UNIVERSIDAD CARLOS III DE MADRID
ESCUELA POLITÉCNICA SUPERIOR

Departamento de Informática

INGENIERÍA TÉCNICA EN INFORMÁTICA DE GESTIÓN

BÚSQUEDAS EN REDES JERÁRQUICAS

Autor: Mónica Escribano Romero
Tutora: Jessica Rivero Espinosa

Abril 2010

No estaría bien dejar pasar la oportunidad de mencionar a todos aquellos que han tenido la paciencia e ilusión de ver en mí una Ingeniera Técnica Informática.

En primer lugar a mis padres: Julio e Isabel, por su gran esfuerzo, su apoyo económico y su gran ayuda a la que gracias a ellos he podido obtener esta titulación. Además, han tenido mucha paciencia, especialmente en esta etapa universitaria, felicitando cada uno de mis logros y dándome ánimos en todo momento para que no me desanimara.

A mi hermano Pedro, por aguantarme en mis momentos con mal carácter, por su paciencia y por su ayuda que he tenido en todo momento aunque no hemos coincidido mucho a lo largo de esta carrera.

A mi novio Carlos, por su apoyo y comprensión tanto a lo largo de la carrera como durante este tiempo que ha durado el proyecto y también por su ayuda, que en muchos casos ha sido buena para avanzar.

A mi tutora de proyecto, por esa gran idea que ha tenido para el desarrollo de éste. Y por la maestría con la que ha dirigido mi proyecto y su cercanía desde siempre.

A todos ellos y a los no mencionados, mil gracias por haberme acompañado en este duro camino, habría sido imposible llegar hasta aquí sin todos vosotros.

Índice

1. Introducción y Objetivos

La localización es una parte inherente de los datos de la empresa: las organizaciones mantienen listas con las direcciones de los clientes, poseen propiedades, envían mercancía desde/hacia los almacenes, gestionan los flujos de mano de obra del transporte, y realizan muchas otras actividades. La mayoría de estas actividades implican la gestión de la ubicación de los diferentes tipos de entidades, incluidos los clientes, propiedades, bienes y empleados. Esos lugares no tienen por que ser estáticos –de hecho, continuamente pueden cambiar con el tiempo. Por ejemplo, los productos son fabricados, envasados, y dirigidos a los almacenes y a los destinos de los clientes, y éstos pueden tener diferentes ubicaciones en las distintas etapas de la red de distribución.

Esto ilustra la omnipresencia de la ubicación o la información espacial en el día a día del negocio. De hecho, las investigaciones estimadas del mercado indican que la mayoría de los datos manejados por las organizaciones tienen asociada, al menos, una dimensión espacial. La capacidad para gestionar adecuadamente el "dónde", o la información espacial, es la clave para la eficacia de las organizaciones, y puede traducirse en importantes ahorros de costes y de mejora de la competitividad comercial. Por ejemplo, la salud, las telecomunicaciones, y las organizaciones gubernamentales locales dependen de la información espacial para ejecutar sus actividades diarias. Otras organizaciones en ámbitos de distribución al por menor, y comercialización usan la información espacial para tomar decisiones estratégicas -por ejemplo, elegir la localización de la tienda, tomar decisiones de inversión, examinar la segmentación del mercado, y estudiar el apoyo a los clientes.

La aparición de los servicios de localización e inalámbricos promete añadir la ubicación de cada elemento de información que utilizamos o procesamos. Tecnologías como *RFID (Radio Frequency IDentification)* tienen el potencial de alterar radicalmente la venta al por menor y la distribución mundial, haciendo posible, a bajo costo, localizar y rastrear los elementos individuales, por pequeños que sean. Con estos nuevos desarrollos, la importancia de la localización ha crecido, y es por ello que cada vez es más importante dominar las herramientas que manejan información espacial.

Las herramientas de *Software* para la gestión de información espacial han sido tradicionalmente conocidas bajo el nombre de *Sistemas de Información Geográfica (GIS)*. Estos sistemas son aplicaciones especializadas en almacenar, procesar, analizar y mostrar datos espaciales. Han sido utilizadas en una variedad de aplicaciones, tales como la ordenación del territorio, geomarketing, logística, distribución, red y gestión de servicios públicos y transporte. Sin embargo, hasta hace poco los *GIS* han empleado modelos específicos de datos espaciales y lenguajes de desarrollo de propiedad, que los mantuvo separados de las principales bases de datos corporativas. Esto ha representado un obstáculo para el pleno despliegue del valor añadido de los datos espaciales en las organizaciones.

Como el uso de los *GIS* en las empresas y en el sector público ha crecido en popularidad, algunas de las limitaciones de los *GIS* se han hecho evidentes. Las organizaciones a menudo tienen que tratar con múltiples estándares e incompatibilidades para el almacenamiento de datos espaciales, y tienen que utilizar distintos idiomas e interfaces para analizar los datos. Además, sistemas como el *CRM (Customer Relationship Management)*, *ERP (Enterprise Resource Planning)*, o los sistemas utilizados en logística dependen cada vez más de la integración de la información espacial con todos los otros tipos de información. Esto ha sido a menudo un desafío operacional y técnico que en algunos casos se resolvió de forma manual para obtener información de un sistema y cargarlo en otro para realizar el análisis espacial necesario.

Oracle Spatial tiene un papel importante en el cambio de esta situación. Una vez que los datos espaciales se almacenan en una base de datos Oracle, pueden ser tratados, recuperados, y relacionados con todos los otros datos almacenados en la base de datos: información espacial, o la ubicación, no es mas que otro atributo de un objeto comercial. Esto elimina la necesidad de coordinar las múltiples fuentes de datos debido a la dependencia de una aplicación, en especial, de las estructuras de datos y el uso de distintos idiomas para consultar los datos. Las características pertinentes de Oracle Spatial son la capacidad de acceder a datos espaciales a través de declaraciones *SQL*, como cualquier otro contenido de la base, y soporte para estándares industriales de información espacial (*SQL* y *OpenGIS*). Por encima de todo, Oracle Spatial permite aprovechar el valor añadido total de la información espacial, que se convierte en una parte integrada de los activos de información de las organizaciones.

Por todo esto se ha decidido desarrollar este proyecto, cuyo objetivo es buscar los caminos en las redes jerárquicas permitiendo de esta manera no trabajar con redes de gran tamaño, sino trabajar con redes más pequeñas filtradas por su nivel dentro de la jerarquía.

Se va a usar una red para poder modelar una red de carreteras que va a estar situada en un primer nivel, y a su vez se va a modelar una red de calles de una ciudad situada en un segundo nivel.

Se van a tener dos niveles porque la red va a ser jerárquica, es decir, va a ver un punto (este caso la ciudad de Fuenlabrada) al que van a estar conectados otros puntos (en este caso puntos de la ciudad de Fuenlabrada, como por ejemplo puntos de interés).

La granularidad (espacio entre los nodos) que se ha definido es la siguiente: Para los puntos en el nivel mas alto va a ser en kilómetros y para los puntos en el nivel mas bajo, es decir, dentro de la ciudad va a ser en metros.

Además, los usuarios que accedan a la aplicación van a estar recogidos en la base de datos que se va a crear, teniendo de antemano la posición en la que se encuentran sin necesidad de hallarla ya que no es algo importante para este proyecto sino mas bien como una línea futura.

Los usuarios que se han creado para esta aplicación han sido tres:

- Un usuario llamado Pepe situado en el nivel 1 (nivel inferior).
- Un usuario llamado Juan situado en el nivel 0 (nivel superior).
- Un usuario llamado Lola situado también en el nivel 0 (nivel superior).

2. Estado del Arte

Para comprender las diferentes tomas de decisiones a lo largo del proyecto así como tener claros los distintos conceptos involucrados en el mismo, es necesario primeramente explicar una serie de conceptos que servirá como introducción a las bases de datos espaciales, que es en lo que se basa este proyecto.

2.1 Uso de Información Espacial en Diversas Industrias

La siguiente lista resume algunos de los principales usos de los datos espaciales, análisis y visualización en diversas industrias.

- Banca y Finanzas: Estas industrias utilizan los datos de localización para el análisis de redes de venta al por menor y de inteligencia de mercado. La base de datos de los clientes combinados con información demográfica ayuda a los bancos a definir una red de venta óptima y a definir la mejor mezcla de productos para ofrecer en cada rama.
- Telecomunicaciones: El análisis de situación ayuda a los operadores de telecomunicaciones y a las compañías a mejorar su posición competitiva. Los datos pueden utilizarse para la planificación de la red, ubicación, organización de mantenimiento, call-center y soporte al cliente, marketing e ingeniería.
- Gobierno local y central: La información espacial es utilizada por todos los organismos gubernamentales, ya que gestionan una gran cantidad de activos distribuidos en grandes territorios. Las aplicaciones incluyen la gestión de los recursos naturales o la ordenación del territorio, mantenimiento de carreteras, viviendas, gestión de emergencias, y servicios sociales.
- Aplicación de la ley: La información espacial ayuda a los oficiales en funciones operativas, así como en el análisis de la delincuencia y la prevención. La información de ubicación es utilizada por los oficiales de campo para localizar lugares y otros recursos en el campo en tiempo real. Los investigadores utilizan los datos espaciales para el análisis de la delincuencia. Los patrones espaciales de la delincuencia se utilizan para localizar mejor los recursos de la policía y mejorar la prevención.
- Actividades inmobiliarias y de gestión de la propiedad: Los datos geográficos y demográficos se usan para identificar y evaluar las ubicaciones de los puntos de venta, la vivienda o instalaciones. Uso de la tierra, el transporte, y la utilidad de las redes se utilizan para zonas industriales y las instalaciones de producción.
- Venta al por menor: Los datos de localización sirven como base para las decisiones operativas y estratégicas. Puede ser utilizado para identificar el perfil de los mejores clientes y ayudar a alcanzar las perspectivas similares. Los datos espaciales pueden incrementar la relevancia y el enfoque de las campañas de marketing, y encontrar el mejor diseño de una red de distribución para el máximo beneficio.
- Utilidades: Muchos sistemas de servicios públicos diferentes pueden ser encontrados en casi todas las calles. Las empresas de servicios públicos utilizan la información espacial para el diseño de estos sistemas de metro, el plan y la base del monitor, y mantener sus redes de cable y tubería.
- Comunicaciones, medios, y publicidad: Los datos de localización se utilizan con frecuencia para aumentar el rendimiento de las campañas de comunicación. La segmentación y ubicación de la focalización basada en ayudar a las compañías figura el momento y la adecuación de las campañas de marketing, lo que aumenta sus expectativas de éxito.

- Los servicios inalámbricos de datos: Cada vez más, los datos de los servicios inalámbricos usan los datos de localización para enriquecer la experiencia del usuario y proporcionar valiosos servicios. Las aplicaciones incluyen los sistemas de navegación personal, buscadores de amigos, de emergencia en carretera, basados en la localización de búsquedas de páginas amarillas, etc. Los servicios de localización inalámbrica son necesarios para un retorno rápido de la inversión hecha en las redes de telecomunicaciones de tercera generación.

2.2 Fuentes de Datos Espaciales

La mayoría de los conjuntos de datos descritos anteriormente son dinámicos, o por lo menos en cierta medida. Sin embargo, hay varios casos en que la razón para utilizar datos espaciales es precisamente su dinamismo. Por ejemplo, el uso de la localización en tiempo real es cada vez más común, gracias al uso generalizado del *GPS* y el uso creciente de sistemas de localización, como la localización *Wi-Fi* o *RFID*, para localizar personas u objetos.

Esto se está usando cada vez más en el caso de la venta al por menor y la distribución de las industrias, en el uso de *RFID*, en lugar de códigos de barras, que hace posible el seguimiento de grandes cantidades de mercancías de forma automática, mientras viajan a través de la distribución de los proveedores en forma de cadena a los usuarios. El etiquetado con *RDIF* puede ser implantado a nivel de los distintos elementos, productos, o incluso de los documentos. Con *RFID*, los bienes pueden seguirse con precisión –por ejemplo, dentro de un almacén- y esta información se puede utilizar para minimizar el inventario, optimizar los horarios de suministro, y crear una oportunidad única de enlazar logísticas con administrativo, *CRM*, y sistemas *ERP*.

2.3 Manejo y Análisis de Datos Espaciales

En esta sección, se examina cómo manejar los datos espaciales y el análisis de cuáles son las funciones típicas de los datos espaciales. Primero, se describe el procesamiento espacial utilizando una terminología genérica.

Las operaciones espaciales suelen incluir:

- Almacenamiento de datos espaciales: En la mayoría de los casos, se trata de lo siguiente:

    - Almacenamiento de los datos de una forma apropiada en la base de datos. Por ejemplo, el sistema de base de datos podría tener un tipo de geometría para almacenar información espacial como puntos, líneas, polígonos y otros tipos de representaciones. El sistema también puede tener un tipo de red para el modelado de las redes de carreteras.
    - Inserción, eliminación y actualización de estos tipos de datos espaciales en la base de datos.

- Análisis de datos del vector espacial: Esto normalmente incluye la funcionalidad de análisis mediante las siguientes operaciones:

    - Dentro de la distancia: Identifica todos los datos espaciales dentro de una distancia determina de un lugar de consulta.
    - Contiene: Identifica todos los datos espaciales que contienen un lugar de consulta especificado (geometría). También se pueden definir funciones para detectar otros tipos de relaciones.

  - Vecino más cercano: Identifica todos los datos espaciales más cercanos a un lugar de consulta.
  - Distancia: Calcula la distancia entre dos objetos espaciales.
  - Buffer: Construye zonas de buffer alrededor de los datos espaciales.
  - Superposición: Superpone diferentes capas de datos espaciales.
  - Visualización: Presenta datos espaciales a través de mapas.

- Análisis de los datos de la red: Por lo general, la mayor parte de los datos espaciales, tales como redes de carreteras, pueden también representarse como una red de datos. Se puede realizar el análisis sobre tales datos mediante la proximidad de la red en lugar de la proximidad espacial.

2.3.1 Almacenamiento de Datos Espaciales en una Base de Datos

En cuanto a los datos, se distingue entre:

- Puntos, cuya descripción espacial requiere sólo coordenadas X e Y (ó X, Y, Z, si se considera 3D).
- Líneas, cuya descripción espacial requiere de unas coordenadas de inicio y fin.
- Polígonos, que se describen por medio de líneas cerradas.

La mayoría de las bases de datos espaciales utilizan un tipo de dato espacial para almacenar datos espaciales denominado geometría. Los usuarios pueden agregar columnas de un tipo de geometría a una tabla de la base de datos para almacenar los objetos espaciales. También hay que destacar que cada columna geométrica puede contener cualquier mezcla de objetos espaciales válido (puntos, líneas, polígonos), y también que cada tabla puede contener una o más columnas geométricas.

La estructuración de los datos espaciales en las tablas y la definición de la estructura de las tablas son las primeras actividades lógicas de cualquier análisis espacial. Afortunadamente, en la mayoría de los casos es una correspondencia intuitiva entre los datos y la estructura de la tabla utilizada para almacenarlos. Sin embargo, en algunos casos se puede encontrar que la base de datos espacial a diseñar puede ser una actividad compleja. Los diseños adecuados pueden facilitar enormemente el análisis, mientras que los pobres pueden hacer que el análisis sea complejo y lento.

2.4 Beneficios de Oracle Spatial

En los últimos 5 a 10 años, los vendedores de bases de datos como Oracle han avanzado en el área espacial. En concreto, Oracle presenta Oracle Spatial para apoyar el procesamiento espacial dentro de una base de datos de Oracle.

Oracle Spatial es la respuesta a la evolución de la tecnología de la información espacial y al papel de los datos espaciales en las principales soluciones IT. *GIS* tiene amplias capacidades para el análisis espacial, pero han desarrollado históricamente sistemas autónomos de información. La mayoría de los sistemas todavía emplean algún tipo de arquitectura dual, con un almacenamiento de datos dedicado a los objetos espaciales y algunos de sus atributos. Esta elección era legítima cuando las principales bases de datos fueron incapaces de manejar eficientemente los datos espaciales. Sin embargo, se ha traducido en la proliferación de formatos de datos propietarios que son tan comunes en la industria de la información espacial. Las consecuencias no deseadas son el aislamiento de *GIS* desde la corriente principal de IT y la creación de la automatización para el procesamiento espacial, frecuentemente desconectados de los servicios centrales de la función de organizaciones IT. Mientras

que las capacidades de *GIS* son ahora impresionantes, los datos espaciales pueden seguir siendo subutilizados, inaccesibles o no se comparten.

Dos acontecimientos principales han cambiado esta situación: la introducción de estándares abiertos de datos espaciales y la disponibilidad de Oracle Spatial. Dos de los estándares abiertos más relevantes son la especificación de características simples *OpenGIS* y *SQL/MM Part 3*. El propósito de estas especificaciones es definir un esquema estándar *SQL* que soporte el almacenamiento, recuperación, consulta y actualización de los datos espaciales a través de una *interfaz de programación de aplicaciones (API)*. A través de estos mecanismos, cualquier otro sistema OpenGIS-compliant o SQL/MM-compliant puede recuperar datos de una base de datos basado en unas especificaciones. Oracle Spatial proporciona una implementación de esas normas y ofrece una forma simple y eficaz de almacenar y analizar datos espaciales dentro de la misma base de datos utilizada para cualquier tipo de datos.

La combinación de estos dos acontecimientos significa que los datos espaciales pueden ser tratados, recuperados, y relacionados con todos los otros datos almacenados en las bases de datos corporativas a través de múltiples fuentes. Esto ha eliminado el aislamiento de los datos espaciales de los procesos de información general de una organización.

Los beneficios de utilizar Oracle Spatial se resumen a continuación:

- Elimina la necesidad de las arquitecturas duales, como todos los datos pueden ser almacenados de la misma forma. Un almacenamiento de datos unificado significa que todos los tipos de datos (texto, mapas y multimedia) se almacenan juntos, en lugar de que cada tipo se almacene por separado.
- Se utiliza *SQL*, un lenguaje estándar para acceder a bases de datos relacionales, eliminando así la necesidad de idiomas específicos para manejar los datos espaciales.
- Se define el tipo de datos SDO_GEOMETRY, que es esencialmente equivalente a los tipos espaciales en los estándares *OGC* y SQL/MM.
- Se implementan formatos SQL/MM “bien conocidos” para especificar los datos espaciales. Esto implica que cualquier solución que se adhiere a las especificaciones de *SQL/MM* pueden almacenar con facilidad los datos en Oracle Spatial y viceversa, sin la necesidad de convertidores.
- Es el estándar de-facto para almacenar/acceder a los datos en Oracle y el intercambio de datos entre aplicaciones por muchos fabricantes, incluyendo *NAVTEQ*, *Tele Atlas*, *Autodesk*, *MapInfo*, *ESRI*, *Bentley*, *Intergraph*, *Radius*, *Skyline* y muchas otras.
- Proporciona escalabilidad, integridad, seguridad, capacidad de recuperación y características avanzadas de administración de usuarios para el manejo de datos espaciales que son la norma en bases de datos Oracle, pero no necesariamente es así en otras administraciones de herramientas espaciales.
- Se elimina la necesidad de organizaciones independientes para mantener una infraestructura de datos espaciales (*hardware*, *software*, soporte, etc.), y se elimina la necesidad de herramientas y habilidades especificas para el funcionamiento de los datos espaciales.
- A través del servidor de aplicaciones, permite a casi cualquier aplicación beneficiarse de la disponibilidad de la información espacial y la inteligencia, reduciendo los costes y la complejidad de las aplicaciones espaciales.
- Con Oracle 10g (y sus posteriores versiones), se introducen los beneficios de la computación en red para bases de datos espaciales. Para las grandes organizaciones que administran los activos de datos muy grandes, tales como cámaras de compensación o servicios públicos, la flexibilidad y escalabilidad de la red puede suponer un ahorro sustancial de costes y facilitar el mantenimiento de las estructuras de base de datos.

- La visualización de gran alcance de los datos espaciales se introduce, eliminando la necesidad de contar con distintas herramientas para muchas aplicaciones.

2.5 Información General de Oracle Spatial

El paquete de tecnología espacial en Oracle permite el almacenamiento de datos espaciales en la base de datos y facilita diferentes tipos de análisis sobre ellos.
A lo largo de este apartado se detallará:

- La funcionalidad de los diferentes componentes del paquete de tecnología espacial en Oracle, incluyendo, una breve introducción al tipo de datos que almacena los datos espaciales (SDO_GEOMETRY) ya que se explicará con más detalle más adelante.
- La descripción de cómo ésta funcionalidad está empaquetada en diferentes productos que se comercializan con diferentes ediciones de software de Oracle.

2.5.1 Tecnología y Arquitectura General

La tecnología de Oracle Spatial se distribuye en dos niveles: el Servidor de Base de Datos y el Servidor de Aplicaciones. La Imagen 1 muestra los distintos componentes que conforman la pila de la tecnología espacial de Oracle y se indica la distribución de los componentes en el Servidor de Base de Datos y los niveles de Aplicación del Servidor.

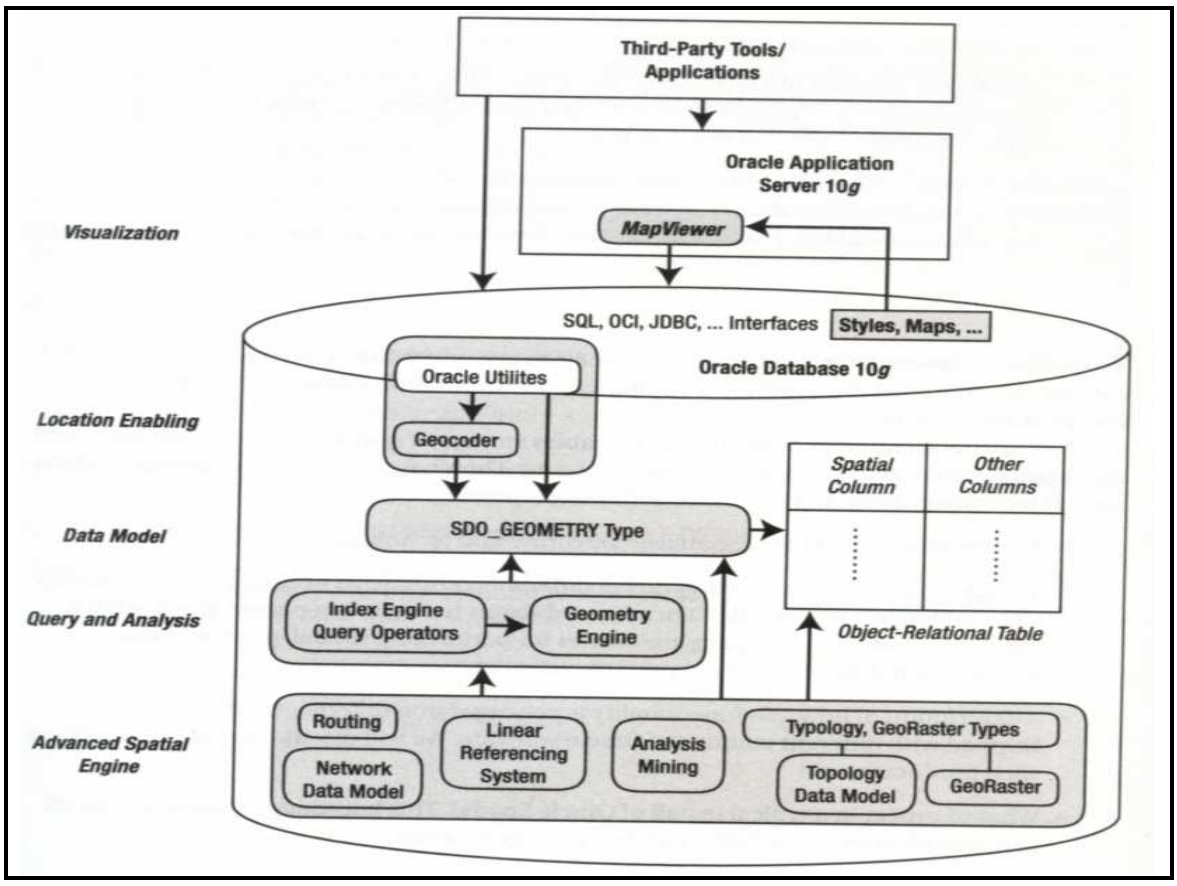


Imagen 1. Componentes de la tecnología de Oracle Spatial

Los componentes básicos de la Imagen 1 son los siguientes:

- Data Model: Oracle Spatial utiliza un tipo de datos *SQL*, SDO_GEOMETRY, para almacenar los datos espaciales dentro de una base de datos de Oracle. Los usuarios pueden definir tablas con columnas de tipo SDO_GEOMETRY para almacenar la ubicación de clientes, tiendas, restaurantes, etc., o los lugares y las extensiones espaciales de las entidades geográficas, tales como calles, carreteras y parques. Este tipo de datos se describe en detalle más adelante.
- Location enabling: Los usuarios pueden añadir columnas SDO_GEOMETRY a las tablas de la aplicación, de manera que se puedan rellenar con los datos de SDO_GEOMETRY utilizando utilidades estándar de Oracle como *SQL*Loader*, importar y exportar. Alternativamente, los usuarios pueden convertir la información espacial implícita, tales como direcciones de calles, en columnas SDO_GEOMETRY utilizando el componente codificador geográfico de Oracle Spatial.
- Spatial Query and analysis: Los usuarios pueden consultar y manipular los datos SDO_GEOMETRY mediante la consulta y el componente de análisis, comprendiendo del Index y Geometry Engines.
- Advanced Spatial Engine: Este componente abarca varios componentes que se adaptan a sofisticadas aplicaciones espaciales, como el *GIS* y la *bioinformática*. Esto incluye, por ejemplo, el componente *GeoRaster* que permite el almacenamiento de objetos espaciales utilizando imágenes (grupos de píxeles) en vez de puntos, líneas y vértices.
- Visualization: Los componentes del servidor de aplicaciones de la tecnología espacial de Oracle incluyen los medios para visualizar los datos espaciales a través de la herramienta de *MapViewer*. *MapViewer* hace que los datos espaciales que se almacenan en las columnas de tipo SDO_GEOMETRY de las tablas de Oracle se puedan mostrar como mapas.

Hay que tener en cuenta también en la Imagen 1 que el componente “Third-Party Tools” puede acceder a datos espaciales, ya sea a través del servidor de aplicaciones o directamente de la base de datos con *SQL*, *OCI*, *JDBC*, o interfaces adecuadas.

2.6 Introducción a Oracle Spatial

La tecnología Oracle Spatial se instala automáticamente con el Standard o Enterprise Edition de un servidor de base de datos de Oracle en sus versiones 10g o posteriores.

2.6.1 Data Model: Almacenamiento de Datos Espaciales

Se puede decir que la información espacial se especifica mediante dos componentes: un lugar con respecto a un origen y una forma geométrica. Es decir:

- La localización específica donde se encuentran los datos con respecto a dos, tres o cuatro coordenadas en el espacio. Por ejemplo, el centro de España está situado en las coordenadas (40.4170, -3.7035) en dos dimensiones “latitud, longitud” del espacio.
- La forma específica de la estructura geométrica de los datos. Punto, línea y polígono son ejemplos de las formas posibles. Por ejemplo, el centro de España está ubicado en las coordenadas (40.4170, -3.7035) en dos dimensiones “latitud, longitud” del espacio y la forma es un punto. Hay que tener en cuenta que el punto especifica tanto un lugar como una forma predeterminada. Alternativamente, se puede especificar la forma de una línea o un polígono de

múltiples puntos de conexión (especificado por su ubicación). Por ejemplo, los límites de la ciudad de Madrid podrían ser un polígono conectado por múltiples puntos.

En algunas aplicaciones, las formas podrían ser más complejas y podrían tener varios polígonos, y/o polígonos que contienen agujeros. Por ejemplo, los limites del estado de Texas y California incluye múltiples polígonos y algunos con islas. En general, la información espacial, en *GIS*, *CAD/CAM*, o una simple localización en aplicaciones, podría ser arbitrariamente compleja.

El tipo de datos SDO_GEOMETRY captura la información de la ubicación y la forma de filas de datos en una tabla. Este tipo de datos se representa internamente como un objeto de tipo de datos Oracle. Se pueden modelar diferentes formas, tales como puntos, líneas, polígonos y las combinaciones adecuadas de cada uno de estos. En resumen, se pueden modelar datos espaciales que ocurren en la mayoría de las aplicaciones espaciales y están conformes con el modelo geográfico *Open GIS Consortium (OGC).*

2.6.2 Location Enabling

Se pueden crear tablas con columnas de tipo SDO_GEOMETRY para almacenar lugares. Por ejemplo se puede crear la tabla us_restaurants_new como se muestra en el Ejemplo 1.

```
SQL> CREATE TABLE us_restaurants_new
(
  id
                                    NUMBER,
  poi_name                                          VARCHAR2(32),
  location                          SDO_GEOMETRY  -- Nueva columna para
almacenar las localizaciones
)
```

Ejemplo 1. Creación de la tabla us_restaurants_new

Desde que SDO_GEOMETRY es un tipo de objeto, como cualquier otro tipo de objeto, se puede rellenar la columna SDO_GEOMETRY utilizando el constructor de objeto correspondiente. Por ejemplo, se puede insertar un lugar de coordenadas (-87, -78) para un restaurante de Pizza Hut en la tabla de us_restaurants_new como se muestra en el Ejemplo 2.

```
SQL> INSERT INTO us_restaurants_new VALUES
(
  1,
  'PIZZA HUT',
  SDO_GEOMETRY
  (
    2001,                                   -- Atributo SDO_GTYPE: "2" el 2001 especifica que
la dimensión es 2.
    NULL,                                                                    --
Otros campos se establecen a NULL.
    SDO_POINT TYPE-- Especifica las coordenadas de un punto
    (
      -87                                   ,
                -- Primera coordenada, por ejemplo, el valor es la longitud
      -78,                                                                   --
Segunda coordenada, por ejemplo, el valor es la latitud
      NULL                                                                   --
Tercera coordenada, si la tiene
    ),
    NULL,
    NULL
  )
);
```

Ejemplo 2. Inserción de un Valor para la Columna SDO_GEOMETRY en una Tabla de Oracle

El objeto SDO_GEOMETRY es instanciado usando el constructor objeto. En este constructor, el primer argumento, 2001, especifica que se trata de un punto geométrico de dos dimensiones, (una línea estaría representada por 2002, un polígono por 2003, y una colección por 2004).

El cuarto argumento almacena la localización de este punto en el atributo SDO_POINT usando el constructor SDO_POINT_TYPE. Aquí, almacena las coordenadas geográficas (-87, -78). En este ejemplo, el resto de argumentos se establecen a NULL.

Hay que tener en cuenta que el Ejemplo 2 muestra una única instrucción *SQL*, INSERT. La carga de datos también se puede realizar a granel con las utilidades de Oracle como *SQL*Loader*, importar/exportar, o interfaces de programación, tales como *OCI*, *OCCI*, y *JDBC*. Estas utilidades y las interfaces se convierten en muy útiles al rellenar los datos desde vendedores *GIS* o proveedores de datos.

En algunas aplicaciones, la información espacial no está de forma explícita como las coordenadas, en cambio, los datos de direcciones de objetos suelen ser la única información espacial disponible. Se pueden convertir los datos de una dirección en un objeto SDO_GEOMETRY utilizando el componente codificador geográfico (siempre con la opción espacial). El codificador geográfico tiene una dirección postal, consultas internas nacionales de base de datos de direcciones y lugares, y calcula la longitud y la latitud para la dirección especificada. Este proceso se conoce como geocodificación en la terminología espacial. La base de datos de dirección/localización suele ser proporcionada por terceros (vendedores de datos). Facilitan esos datos para Estados Unidos, Canadá y Europa, *NAVTEQ* y *Tele Atlas*.

El Ejemplo 3 muestra como utilizar el geocoder para obtener las coordenadas en forma de un objeto SDO_GEOMETRY para la dirección '3746 CONNECTICUT AVE NW'en Washington, D.C.

```
SQL> SELECT
SDO_GCDR.GEOCODE_AS_GEOMETRY
(
  'SPATIAL'
                -- Esquema espacial geocoder del almacenamiento de datos
  SDO_KEYWORDARRAY                -- Objeto combinado de diferentes componentes de
dirección
  (
    '3746 CONNECTICUT AVE NW',
    'WASHINGTON, DC 20008'
  ),
  'US'
                                                        -- Nombre del país
) geom.
FROM DUAL;
```

```
GEOM (SDO_GTYPE, SDO_SRID, SDO_POINT (X, Y, Z), SDO_ELEM_INFO, SDO_ORDINATES)
 ----------------------------------------------------------------------------------------------------------------------
SDO_GEOMETRY
(
  2001,
  8307,
  SDO_POINT_TYPE (-77.060283, 38.9387083, NULL),
  NULL,
  NULL
)
```

Ejemplo 3. Conversión de Datos de dirección (Información Espacial implícita) al Objeto SDO_GEOMETRY (Información Espacial Explicita)

Esta función de geocodificación, geocode_as_grometry, toma tres argumentos. El primer argumento es el esquema. En este ejemplo, se usa el esquema 'SPATIAL'. El segundo argumento especifica un objeto SDO_KEYWORDARRAY, formado por diferentes componentes de una dirección. En este ejemplo, SDO_KEYWORDARRAY se construye a partir de la calle '3746 Connecticut Ave NW' y la ciudad o el código postal 'Washington, DC 20008'.El tercer argumento especifica el conjunto de datos de 'US' que se utiliza para la codificación geográfica de la dirección especificada. La función devuelve un tipo SDO_GEOMETRY con la longitud -77.060283 y la latitud 38.9387083.

El codificador geográfico también puede realizar una coincidencia aproximada. De la misma manera que los motores de búsqueda pueden buscar palabras relacionadas, así como palabras exactas, Oracle puede realizar una coincidencia aproximada en los nombres de las calles y así sucesivamente. Por ejemplo, si el campo dirección en el Ejemplo 3 está mal escrito como 'CONNETICUT AVE', el codificador geográfico podría realizar una correspondencia aproximada para que coincidan los campos con los errores ortográficos en la base de datos.

Hay que tener en cuenta que el tipo de datos SDO_GEOMETRY es como cualquier otro tipo de objetos en la base de datos. Los usuarios pueden ver los datos, examinar y modificar los atributos. En contraste, varios proveedores de datos de *GIS* y los socios tienen sus propios formatos binarios para la representación de la información espacial. Estos fabricantes suelen proporcionar herramientas para la carga de datos o la conversión de los datos en formatos estándar de Oracle.

2.6.3 Query and Analysis

La consulta y el análisis proporcionan la funcionalidad básica para la investigación y el análisis de las geometrías espaciales. Este componente tiene dos subcomponentes: un Motor de Geometría y un Motor de Índice. A través de estos componentes se llevan a cabo las consultas y el análisis espacial.

El Motor de Geometría

El motor de geometría proporciona funciones para analizar, comparar y manipular geometrías. Por ejemplo, se podría utilizar esta funcionalidad para identificar la geometría más cercana a los cinco restaurantes de I-795 en Washington. El Ejemplo 4 muestra esta operación.

```
SQL> SELECT poi_name
FROM
  (
    SELECT poi_name,
      SDO_GEOM.SDO_DISTANCE (P.location, I.geom, 0.5) distance
    FROM us_interstates I, us_restaurants P
                          WHERE I.interstates = 'I795'
      ORDER BY distance
  )
WHERE ROWNUM <= 5;
```

```
POI_NAME
----------------
PIZZA BOLI'S
BLAIR MANSION INN DINNER THEATER
KFC
CHINA HUT
PIZZA HUT
```

Ejemplo 4. Encontrar los 5 Restaurantes Más Cercanos a I-795

Se puede observa que la cláusula SELECT interna calcula la distancia entre I-795 (que no es una carretera principal) y cada fila "restaurante" de la tabla us_restaurants utilizando la función SDO_GEOM.SDO_DISTANCE del motor de geometría. También, la cláusula ORDER BY ordena los resultados por orden ascendente de la distancia. La instrucción SELECT exterior selecciona las cinco primeras filas, o los cinco restaurantes más cercanos.

En la consulta anterior, la ubicación de la carretera I-795 se compara con cada fila de un restaurante de la tabla, independientemente de que este muy lejos de la carretera I-795. Esto podría significar un tiempo considerable que se gasta en la tramitación de las filas de los restaurantes que están muy lejos de la carretera I-795 y por lo tanto son irrelevantes para la consulta. Para acelerar el procesamiento de las consultas y reducir al mínimo la sobrecarga del procesamiento, es necesario crear índices sobre la ubicación de los restaurantes.

El Motor de Índice

Oracle Spatial proporciona el Motor de Índices espaciales para este propósito. El Ejemplo 5 muestra un ejemplo de cómo crear un índice sobre la ubicación de los restaurantes.

```
SQL> DROP INDEX us_restaurants_sidx;
SQL> CREATE INDEX us_restaurants_sidx ON us_restaurants (location)
INDEXTYPE IS mdsys.spatial_index;
```

Ejemplo 5. Creación de un Índice sobre la Ubicación (Columna SDO_GEOMETRY) de los Restaurantes

El Ejemplo 5 primero elimina el índice que existe. En la segunda y tercera línea, se muestra como crear el índice espacial. Hay que tener en cuenta que la base de datos cuenta con la cláusula INDEXTYPE para crear un índice espacial de la columna "location" de la tabla us_restaurants. Este índice es un índice especializado para atender a los datos SDO_GEOMETRY. El uso de tal índice implica que el Motor de Índice en Oracle Spatial pode las filas del procesamiento de la consulta y por lo tanto se acelere la consulta para la mayoría de las aplicaciones. El Motor de Índice proporciona funciones equivalentes, denominadas operadores, para la identificación de las filas de la tabla que satisfacen una proximidad especificada (como la proximidad a la carretera I-795). Se puede reescribir la consulta anterior para encontrar los cinco restaurantes más cercanos a la carretera I-795 utilizando el índice como base del operador. El Ejemplo 6 muestra la consulta resultante.

```
SQL> SELECT poi_name
FROM us_interstates I, us_restaurants P
WHERE I.interstate = 'I795'
  AND SDO_NN (P.location, I.geom) = 'TRUE'
  AND ROWNUM <= 5;
```

```
POI_NAME
----------------
PIZZA BOLI'S
BLAIR MANSION INN DINNER THEATER
KFC
CHINA HUT
PIZZA HUT
```

Ejemplo 6. Encontrar los Cinco Restaurantes más Cercanos a I-795 Usando el Índice Espacial

Se puede observar que esta consulta devuelve las mismas filas que el Ejemplo 4. Sin embargo, esta consulta tiene una estructura más sencilla, sin subconsultas. Utiliza un nuevo índice basado en el operador llamado 'SDO_NN', con 'NN' como abreviatura para 'Nearest-Neighbor' (vecino más cercano). Este índice se obtendrá del operador de la tabla 'us_restaurants' cuando los datos de la columna 'location' estén cerca de la carretera I-795. El operador SDO_NN devuelve estas filas en orden de proximidad a la carretera I-795. El predicado 'ROWNUM' determina cuantos restaurantes cercanos tienen que ser devueltos en la consulta. En consecuencia, es probable que ésta consulta se ejecute más rápido que la consulta del Ejemplo 4.

2.6.4 Visualizando Datos Espaciales

Para visualizar los resultados de las consultas espaciales, Oracle Technology incluye el componente *MapViewer* para facilitar la generación de mapas de datos espaciales. Cada mapa se asocia con un conjunto de temas. Cada tema denota datos espaciales de una tabla específica, y se asocia con un estilo de representación. Por ejemplo, se puede especificar el tema de las carreteras interestatales (datos de la tabla de carreteras interestatales) como líneas gruesas azules. Oracle Spatial proporciona vistas al diccionario, USER_SDO_MAPS, USER_SDO_THEMES y USER_SDO_STYLES, para definir nuevos mapas, asociarlos a temas y estilos para especificar la prestación de los temas dentro de la base de datos.

Además, *MapViewer* permite asociar un nombre al mapa especificado. Básicamente, un servlet consulta la base de datos y recupera los temas y las normas de estilo asociadas a un nombre de mapa especificado. Usando esta información, el servlet de *MapViewer* genera una imagen del mapa construido. La Imagen 2 muestra un mapa construido con *MapViewer*.

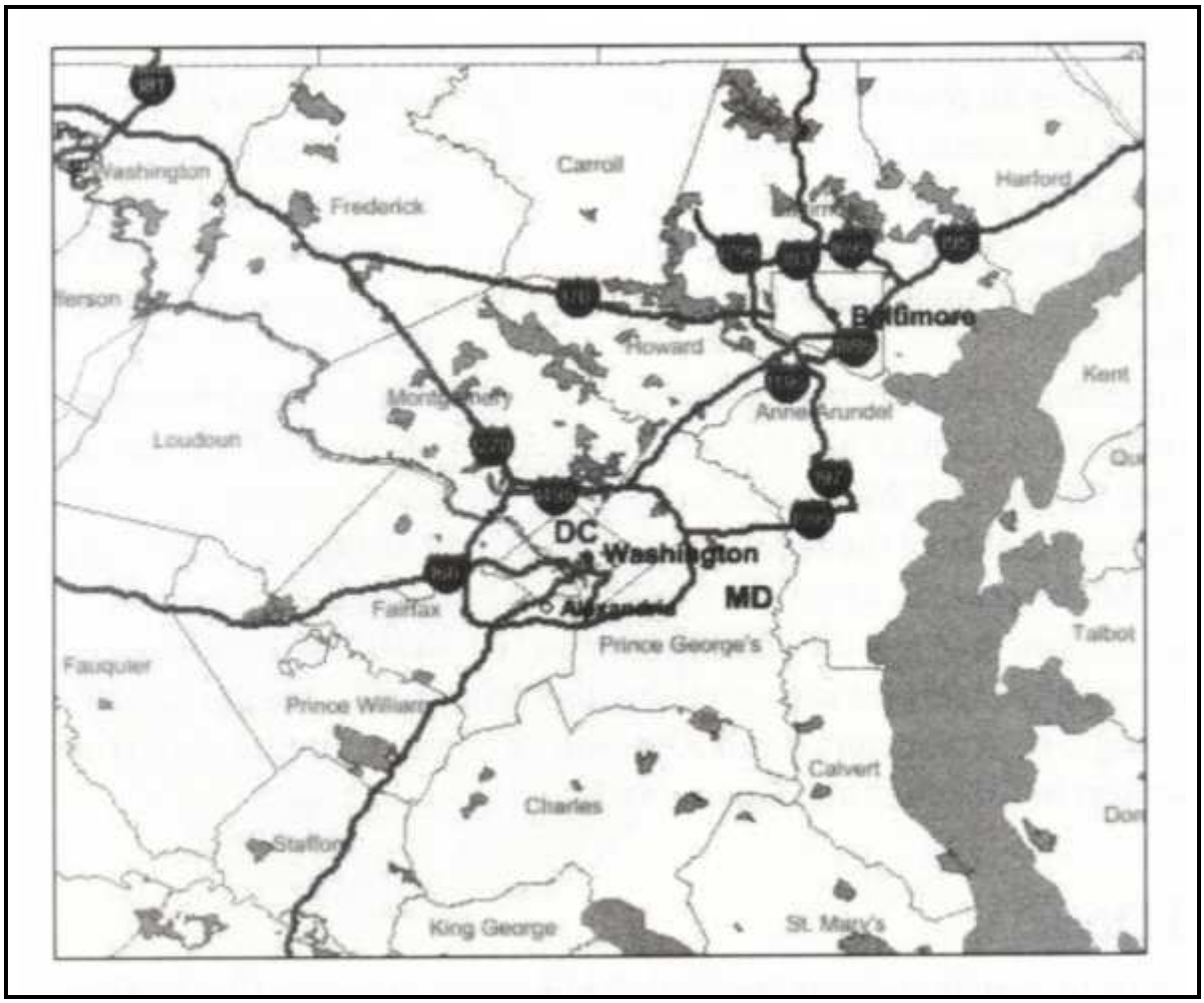


Imagen 2. Ejemplo de mapa con varios temas generado utilizando *MapViewer*.

Dicho mapa se compone de varios temas: ciudades, límites de ciudades, ríos, carreteras interestatales, parques. Las ciudades DC y Baltimore, se representan como puntos en color negro. Los condados, Howard, Fairfax, etc., se muestran como polígonos blancos. El río del lado derecho del mapa se muestra con un color gris oscuro. Las carreteras interestatales, como la carretera I-795, se representan como cadenas de líneas negras, un parque se representa como un polígono en gris claro.

En este mapa, también se puede superponer la localización de los cinco restaurantes más cercanos a la carretera I-795.

Hay que tener en cuenta que el mapa de la Imagen 2 muestra los datos almacenados de diferentes temas en la columna SDO_GEOMETRY de las tablas. Además, *MapViewer* puede mostrar los datos espaciales almacenados en formato de trama (o imagen). Estos datos son almacenados en las tablas de Oracle usando el tipo de datos SDO_GEORASTER.

2.6.5 Advanced Spatial Engine

Advanced Spatial Engine tiene varios subcomponentes que se adaptan al análisis complejo y a la manipulación de datos espaciales que se requieren en las aplicaciones *GIS* tradicionales.

Internamente, cada uno de estos componentes adicionales utiliza el tipo de datos, el índice geométrico y la funcionalidad del motor de geometría. Dichos subcomponentes se listan a continuación:

- El Modelo de Datos de Red proporciona un modelo de datos para el almacenamiento de redes dentro de la base de datos de Oracle. Los elementos de la red (enlaces y nodos) pueden estar asociados con los costes y límites, por ejemplo, el modelo de los límites de velocidad para los segmentos de la carretera. Otra funcionalidad incluye el cálculo de la ruta más corta entre dos localizaciones determinadas de una red de carreteras, la búsqueda de los N nodos más cercanos, y así sucesivamente. El modelo de datos de red es útil en aplicaciones de enrutamiento. Las típicas aplicaciones de enrutamiento incluyen servicios Web tales como *MapQuest* y *Yahoo!*, o aplicaciones de navegación para usuarios móviles mediante tecnología *GPS*.
- El *Sistema de Referencia Lineal (LRS)* facilita la traducción de los marcadores de kilómetros en una autopista (o cualquier otra característica lineal) al espacio de coordenadas geográficas y viceversa. Este componente permite a los usuarios hacer frente a los distintos segmentos de una geometría lineal, como una carretera, sin llegar a referirse a las coordenadas del segmento. Esta funcionalidad es útil en el transporte y en aplicaciones de utilidad, tales como la gestión de gaseoductos.
- El Análisis Espacial y el Motor de Minería proporcionan una funcionalidad básica para combinar el análisis demográfico y espacial. Esta funcionalidad es útil en la identificación de sitios potenciales para el inicio de nuevas tiendas basados en la densidad de clientes e ingresos. Estas herramientas también pueden usarse para materializar la influencia del barrio, que a su vez puede ser utilizado en la mejora de la eficacia y la capacidad de predicción de los datos del Motor de Minería de Oracle.
- *GeoRaster* facilita el almacenamiento y recuperación de imágenes georeferenciadas usando sus huellas espaciales y los metadatos asociados. Define un nuevo tipo de datos para almacenar imágenes de mapas de bits de los objetos con referencia geográfica. Esta funcionalidad es útil en la gestión de las imágenes del satélite.
- La Topología del Modelo de Datos admite un análisis detallado y la manipulación de los datos de la geometría espacial, utilizando elementos topológicos como nodos y bordes. Oracle Spatial define un nuevo tipo de datos para representar los elementos topológicos que pueden ser compartidos entre los diferentes objetos espaciales. Las actualizaciones de elementos compartidos se definen implícitamente en las actualizaciones de la distribución geométrica de los objetos.

2.7 Productos de la Tecnología de Oracle Spatial

En las secciones anteriores, se han descrito brevemente las funcionalidades que Oracle Spatial brinda para apoyar a las siguientes operaciones sobre los datos espaciales:

- Almacenamiento de modelo de datos utilizando el tipo de datos SDO_GEOMETRY.
- Consulta y análisis que utiliza el Motor de Índice y el Motor de Geometría.
- Localización que permite utilizar el codificador geográfico de la conversión de datos de direcciones en datos de tipo SDO_GEOMETRY.
- Visualización usando el *MapViewer*.
- Funcionalidades del Motor Espacial Avanzado tales como el análisis de redes.

MapViewer, la herramienta de visualización espacial, se incluye como parte de Oracle Application Server. El resto de funcionalidad espacial se incluye, a veces, opcionalmente, con el servidor de base de datos.

2.7.1 Locator

Locator proporciona un subconjunto de la funcionalidad de la base espacial para atender a aplicaciones específicas. En concreto, incluye las siguientes funcionalidades:

- El modelo de datos para el almacenamiento de datos espaciales usando el tipo de datos SDO_GEOMETRY: Esto incluye el almacenamiento de todos los tipos de geometrías (puntos, líneas, polígonos, etc.).
- Consultas y análisis que utiliza el Motor de índice: Ésta incluye la creación de índices espaciales y consultas mediante operadores espaciales asociados como SDO_NN.
- El SDO_GEOM.SDO_DISTANCE y las funciones SDO_GEOM.VALIDATE_GEOMETRY_XXX: Éstas y otras funciones son también parte del Locator.

La Imagen 3 muestra la funcionalidad proporcionada en el Locator. Los componentes del Locator se destacan en color negro. Los elementos que pertenecen al Locator se muestran en gris.

Las aplicaciones que utilizan el Locator pueden necesitar de los servicios de geocodificación para convertir las direcciones en tablas de la aplicación. Después de almacenar las localizaciones espaciales como columnas de tipo SDO_GEOMETRY, el Locator permite una gran variedad de consultas espaciales, como la identificación de los clientes en un territorio determinado de ventas o el cajero automático más cercano a una ubicación específica. El Locator se utiliza normalmente en las siguientes aplicaciones:

- Aplicaciones Simple *GIS*, que sólo pueden trabajar con datos geográficos, tales como estados, ciudades, o los límites de la propiedad y el índice de consultas basadas en el uso de operadores espaciales asociados. Normalmente, sin embargo, la mayoría de las aplicaciones de los *GIS* pueden necesitar la funcionalidad del Motor de Geometría (que no es compatible con el Locator).
- Aplicaciones de negocio Simple, donde los datos espaciales se obtienen de los proveedores. El índice base en operadores de apoyo en el Locator puede realizar una gran cantidad de análisis en las aplicaciones empresariales.

- CAD/CAM y aplicaciones similares, donde los datos espaciales no se refieren a los lugares en la superficie de la Tierra. Por ejemplo, en aplicaciones CAD/CAM, los datos representan la estructura y formas de las diferentes piezas de un automóvil. En este caso, los datos están en dos o tres dimensiones en el espacio de coordenadas, es decir, no hay necesidad de convertir las columnas no espaciales (como las direcciones) para obtener información espacial. Las operaciones que son necesarias para aplicaciones como éstas son el índice basado en el análisis de los operadores de proximidad. Las funciones avanzadas de motores espaciales como el enrutamiento no son de utilidad en estas aplicaciones.

  En resumen, el Locator ofrece un subconjunto basado en la tecnología espacial.

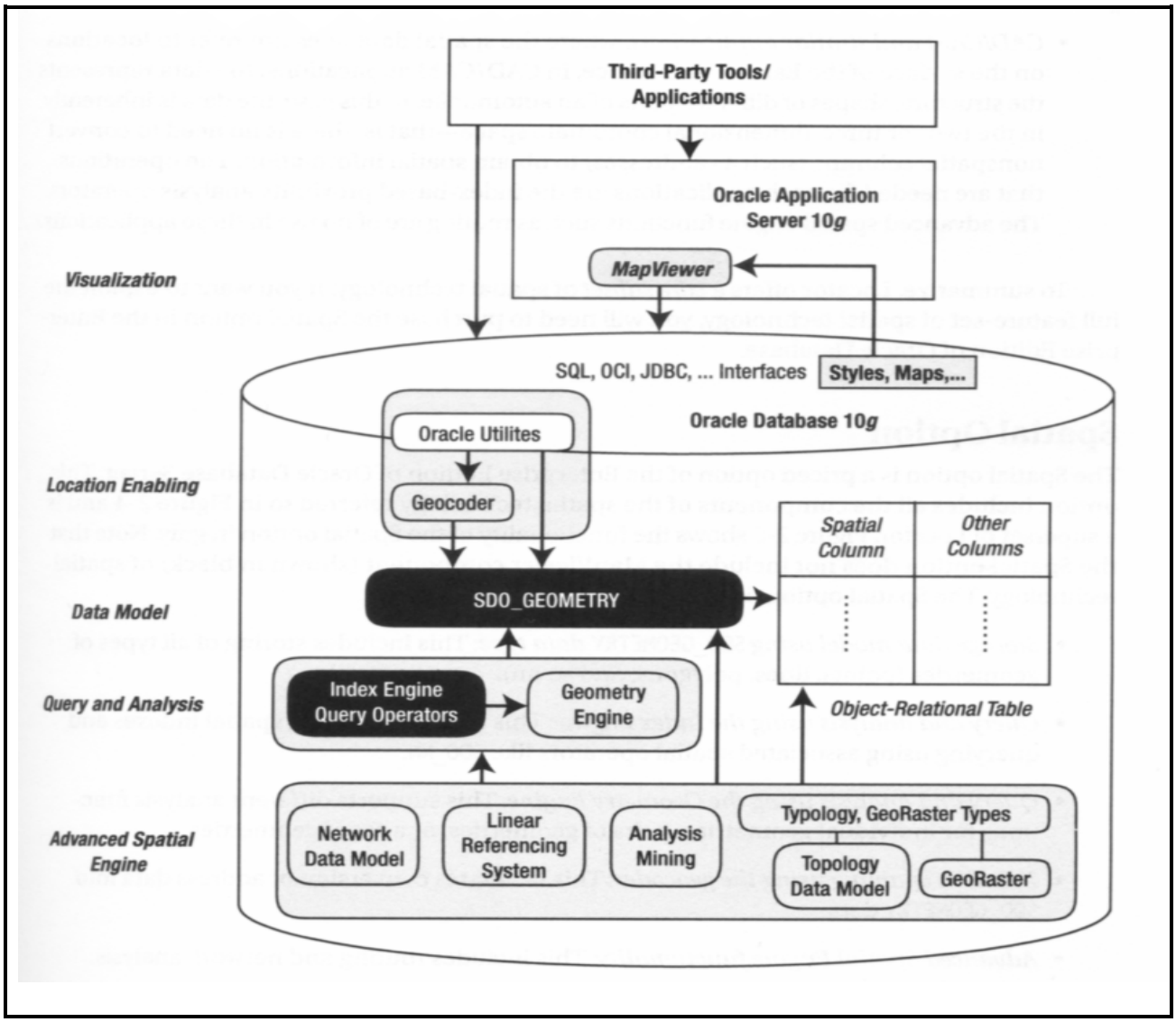


Imagen 3. La funcionalidad del Locator, la parte libre de la tecnología espacial, se muestra en los óvalos negros

2.7.2 Opción Espacial

Esta opción incluye todos los componentes de la tecnología espacial a la que se refiere en la Imagen 3, y es un superconjunto del Locator. La Imagen 4 muestra la funcionalidad de la opción espacial en gris. Hay que tener en cuenta que la opción espacial no incluye el componente *MapViewer* (en negro) de la tecnología espacial. La opción espacial consiste en:

- Almacenamiento del modelo de datos usando datos de tipo SDO_GEOMETRY: Esto incluye el almacenamiento de todos lo tipos de geometrías (puntos, líneas polígonos, etc.).
- Consultas y análisis que utiliza el Motor de Índice: Esta incluye la creación de índices espaciales y consultas mediante operadores espaciales asociados como SDO_NN.
- Consultas y análisis que utiliza el Motor de Geometría: Este es compatible con funciones de análisis diferentes para las geometrías individuales, parejas de geometrías, o conjunto de geometrías.
- La localización permite utilizar el codificador geográfico: Esto facilita la conversión de datos de direcciones en los datos de tipo SDO_GEOMETRY.
- La funcionalidad Avanzada del Motor Espacial: Esto incluye enrutamiento y análisis de redes.

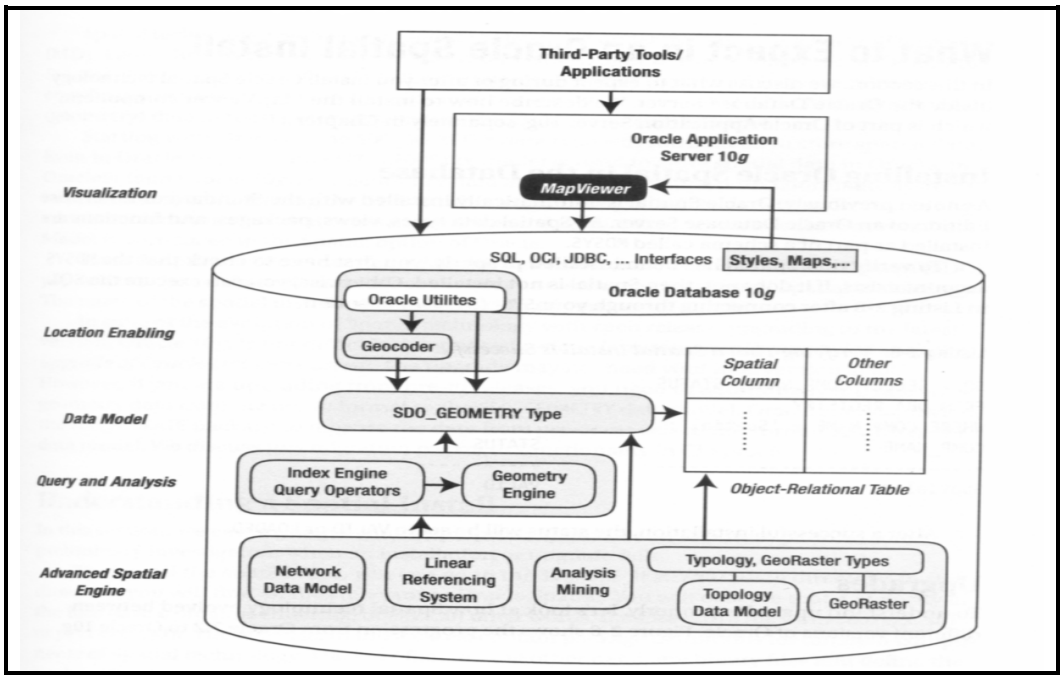


Imagen 4. La funcionalidad de la opción Espacial se muestra en gris.

Una amplia variedad de aplicaciones pueden utilizar el conjunto completo de la funcionalidad proporcionada en la opción espacial.

2.8 Permitir localización a las Aplicaciones

A continuación se describe como incrementar las tablas de aplicaciones existentes con información sobre la localización. Esta información, normalmente, deriva de los componentes de las direcciones en las tablas de aplicaciones como pueden ser clientes, sucursales, competidores, y se almacena en estas tablas como la localización de puntos.

Se puede realizar el análisis, al combinar los datos de la aplicación con los datos geográficos, tales como redes de calles, límites de ciudades, etc. Las redes de calles y límites de ciudades son más complejas que la información de localización en las tablas de las aplicaciones. Estas redes y límites (es decir, los datos geográficos) necesitan ser almacenados como líneas, polígonos, y otros tipos de geometrías complejas. Se describen varias opciones de diseño a considerar durante el almacenamiento de estos datos geográficos en las tablas de Oracle.

Después de configurar las aplicaciones y las tablas de datos geográficos, es necesario insertar metadatos espaciales específicos de la localización para habilitar estas tablas para su posterior análisis.

2.8.1 Añadir Información de localización a las Tablas

La mayoría de los datos de una aplicación pueden ser clasificados en dos tipos de tablas:

- Tablas especificas de la aplicación: Éstas contienen información que es específica para su aplicación. Las tablas de la aplicación utilizarán técnicas de normalización estándar para llegar a un conjunto apropiado de tablas para almacenar los datos. Estas tablas no pueden contener datos espaciales explícitos. Sin embargo, pueden tener información espacial implícita en forma de direcciones.
- Tablas geográficas: Los datos geográficos son independientes de la aplicación y contienen columnas para almacenar información espacial explícita para redes de calles, límites de ciudades, etc. Estos datos pueden ser utilizados como un valor añadido en la aplicación.

La Imagen 5 muestra un ejemplo de estos dos conjuntos de tablas para una aplicación empresarial.

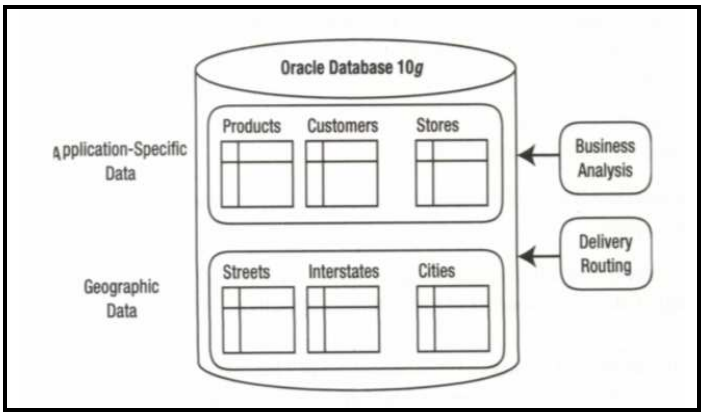


Imagen 5. Datos para una aplicación de ejemplo

Datos específicos de aplicaciones

Para los datos específicos de aplicaciones, las reglas estándar de normalización pueden ser empleadas para diseñar el conjunto de tablas de la aplicación que mejor se adapte a las necesidades de ésta.

Las tablas pueden ser creadas con los atributos adecuados.

En un nivel fundamental, permitir la localización en una aplicación por ejemplo de negocios, requiere guardar la información de localización de los clientes, sucursales, competencia, etc. Esto significa que se necesita incrementar las tablas correspondientes con una columna adicional para almacenar la localización. Esta información de localización básica se almacena como un punto, utilizando el tipo SDO_GEOMETRY.

Esto, por sí mismo, no rellena la columna de localización. Si se selecciona dicha columna de la tabla, sólo mostrará valores nulos.

La manera más común para rellenar las columnas de localización de las tablas de la aplicación es geocodificando las direcciones de las columnas apropiadas. La Imagen 6 ilustra el proceso de geocodificación.

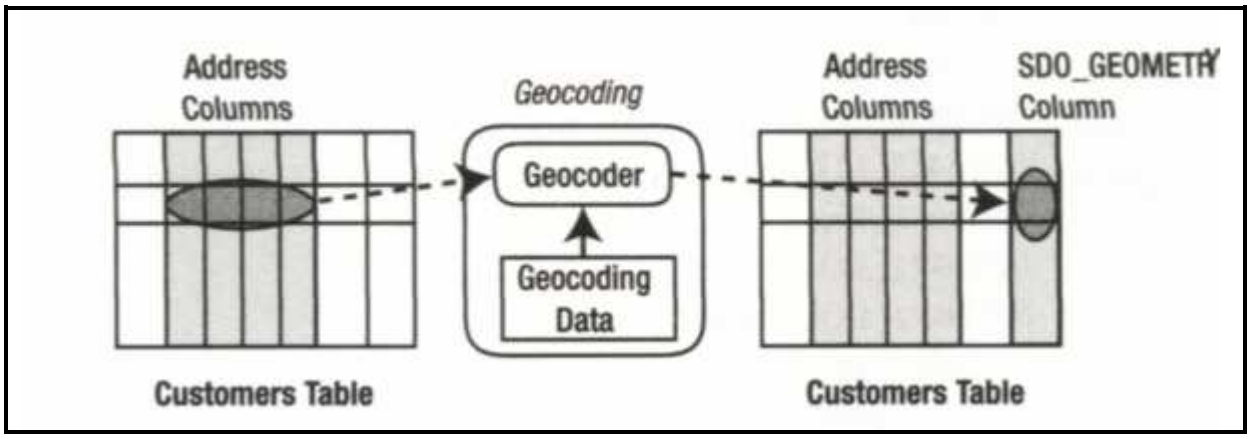


Imagen 6. Geocodificación de datos de la aplicación para rellenar columnas de tipo SDO_GEOMETRY

Como se muestra en la Imagen 6, estas herramientas consultan una base de datos interna para determinar los valores de latitud y longitud para una dirección especificada. Estos pares <longitud, latitud> pueden ser almacenados como un punto geométrico usando el tipo de datos SDO_GEOMETRY. Oracle Spatial proporciona una herramienta de geocodificación para la traducción de direcciones (información espacial implícita) en objetos SDO_GEOMETRY.

Por ejemplo, se puede decir que las tablas de clientes, proveedores, sucursales, competidores almacenarán información de direcciones. Esta dirección se almacena típicamente usando los siguientes atributos: nombre de la calle, número, ciudad, y código postal, todos estos son de tipo VARCHAR2. El Ejemplo 7 muestra la información de la dirección en la tabla clientes para un cliente específico.

```
SQL> SELECT street_number, street_name, city, state, postal_code
FROM customers
WHERE id=1;

134   12TH   St   SE   WASHINGTON   DC   20003
```

Ejemplo 7. Ejemplo de dirección para un Cliente Específico de la Tabla de clientes.

Oracle Spatial permite convertir esta dirección (nombre de la calle, número, ciudad y código postal) en un punto de localización de dos dimensiones en la superficie de la Tierra. La función específica que se necesita es sdo_gcdr.geocode_as_geometry. Esta función toma el nombre del esquema y el nombre del conjunto de datos de códigos geográficos. El segundo argumento es un objeto de tipo sdo_keywordarray construido fuera de los componentes de la dirección nombre de la calle, número, ciudad y código postal.

```
SQL> UPDATE customers
  SET location =
  SDO_GCDR.GEOCODE_AS_GEOMETRY
  (
    'SPATIAL'
    SDO_KEYWORDARRAY
    (
      street_numer  ||  ' ' ||  street_name,
      city  ||  ', '  || state ||  ' '  || postal_code
    ),
    'US'
);
```

Ejemplo 8. Geocodificación de Direcciones para Obtener Información Espacial Explicita

Ahora se puede examinar lo que aparece en la información de localización. Simplemente se selecciona la columna de la tabla de clientes. El Ejemplo 9 muestra el *SQL* que hace esto.

```
SQL> SELECT location
FROM customers
WHERE id=1;
```

```
LOCATION(SDO_GTYPE, SDO_SRID, SDO_POINT(X,Y,Z), SDO_ELEM_INFO, SDO_ORDINATES)
--------------------------------------------------------------------------------------------------------------------------
SDO_GEOMETRY (2001, 8307, SDO_POINT_TYPE(-76.99022, 38.888654, NULL), NULL, NULL)
```

Ejemplo 9. Columna de localización Codificada en la Tabla clientes.

Se observa que la dirección especificada (street_numer='134', street_name='12th ST SE', city='WASHINGTON', y postal_code='20003') se traduce en un objeto SDO_GEOMETRY con los valores de longitud y latitud de -76.99 y 38.889 en el atributo sdo_point (instancia usando el objeto SDO_POINT_TYPE). El valor sdo_gtype de 2001 indica que la localización en de dos dimensiones (2 en 2001) punto (1 en 2001) localización.

Precaución: Las posiciones de coordinas se refieren comúnmente como "latitud/longitud". Sin embargo, en Oracle Espacial, las coordenadas se almacenan como longitud seguida de latitud.

Una vez que se construye un objeto SDO_GEOMETRY, se puede insertar, actualizar, y consultar al igual que cualquier otra columna de una tabla en Oracle.

Consideraciones de Diseño para Datos Específicos de Aplicaciones

Como se señaló, la organización de los datos en las tablas correspondientes depende de la aplicación y probablemente incluirá técnicas de diseño estándar, tales como la normalización, diagrama de modelado basado en entidad-relación (ER), etc. Oracle Spatial no tiene ningún tipo de recomendaciones o restricciones específicas de cómo organizar los datos de las aplicaciones.

Se debe considerar la partición de una tabla cuando ésta tiene millones de filas. Por ejemplo, los datos de los clientes para todo un país. Estos clientes comparten los mismos atributos. Como resultado, la normalización y otras técnicas de modelado pueden recomendar el almacenamiento de todos los clientes en una sola tabla. Sin embargo, para aplicaciones espaciales, el número de clientes puede ser elevado, alcanzando decenas de millones o miles de millones.

La partición puede ayudar a que las tablas grandes sean eficaces y eficientes. Además, también puede mejorar la eficiencia de las operaciones de análisis espacial mediante el uso de la poda cada vez que la clave de partición se especifique en la consulta *SQL*.

2.8.2 Datos Geográficos

Si se quiere visualizar con precisión los lugares almacenados en las tablas de la aplicación en un mapa, entonces hay que mostrar los límites no sólo de las calles y ciudades, sino también de los ríos, parques nacionales, etc.

Estos datos geográficos están generalmente disponibles en una variedad de fuentes, incluidos los distribuidores comerciales de *GIS* y los organismos cartográficos nacionales. *NAVTEQ* y *Tele Atlas* son dos vendedores de estos programas, y ambos venden los datos geográficos de Estados Unidos y Europa. El *Ordnance Survey* es la agencia nacional de cartografía de Gran Bretaña, y suministra una cobertura muy detallada de Gran Bretaña llamada *MapsterMap*.

Para permitir la integración efectiva y el análisis, los datos geográficos, así como los datos específicos de las aplicaciones, deben ser almacenados dentro de la base de datos. Esto significa que hay que ser capaces de almacenar una amplia gama de diferentes tipos de datos. Por ejemplo, una red de calles puede ser representada por un conjunto de líneas de conexión de dos puntos de diferentes dimensiones. Del mismo modo, el límite de una ciudad puede ser representado por un polígono conectado por líneas. Estos tipos de datos espaciales se pueden representar utilizando el mismo tipo de datos SDO_GEOMETRY.

Consideraciones de Diseño para Datos Geográficos

Cada tipo de dato geográfico puede tener los siguientes atributos:

- Estados: los atributos pueden incluir el nombre, la abreviatura del nombre, la población, el ingreso promedio de las horas, la frontera (ésta última se almacena en un objeto SDO_GEOMETRY).
- Condados: los atributos pueden incluir el nombre, el nombre del estado al que pertenece, la superficie, la población por kilómetro cuadrado, y un objeto SDO_GEOMETRY que almacena la frontera del condado.
- Interestatales: los atributos pueden incluir el nombre y un objeto SDO_GEOMETRY para almacenar la forma lineal interestatal.
- Calles: los atributos pueden incluir el nombre, la ciudad, el estado, y un objeto SDO_GEOMETRY para almacenar la forma lineal de la calle.

El almacenamiento de las calles, carreteras interestatales, los condados y los estados en una sola tabla es probable que sea ineficiente (ya que puede ralentizar el análisis posterior) y este efecto debe evitarse. Para ello se deben almacenar estos datos en diferentes tablas siguiendo los criterios generales que se muestran a continuación:

1. Separar los datos espaciales que no comparten los mismos atributos: Esto es similar a las técnicas utilizadas para la normalización de los datos regulares. Por ejemplo, los estados tendrán diferentes atributos que los datos de los condados, calles o carreteras interestatales.
2. Separar los datos más gruesos de los más finos: las calles y las carreteras interestatales representan formas lineales. A veces, incluso pueden compartir el mismo conjunto de atributos. Sin embargo, las carreteras pasan a través de múltiples estados, mientras que las calles están localizadas en una determinada ciudad o región. Como el número de calles es probable que sea mucho mayor que el número de carreteras, el almacenamiento de las calles y carreteras en una misma tabla puede causar problemas de rendimiento cuando el acceso sea sólo a los datos de las carreteras.
3. Separar la forma de la geometría: Si se separan los datos espaciales basados en la forma geométrica –en otras palabras, en función de si se trata de un punto, una línea o un polígono- entonces se pueden utilizar los mecanismos del control de tipo previstos por los índices de Oracle Spatial en el tiempo de inserción. Por ejemplo, si se ha creado un índice espacial y se precisa de una tabla en la que solo había puntos, el índice produciría un error si se encontró con una geometría distinta de puntos en la tabla. Los índices espaciales pueden funcionar mejor si saben qué tipo de geometría se espera en la tabla.
4. Partición de datos localizados: Por ejemplo, los datos de las calles para todo un país. Las calles parten de los mismos atributos y también están en el mismo nivel de resolución. Basándonos en los últimos tres criterios, se pueden almacenar todas las calles en la misma tabla. Sin embargo, debido al gran número de filas de dicha tabla, la aplicación puede beneficiarse de la partición de la misma.

Además de separar los datos específicos de las aplicaciones y los datos geográficos en tablas, también es necesario especificar la información adicional llamada metadatos para permitir la localización en la aplicación. Estos metadatos se utilizan en una variedad de funciones espaciales, tales como la validación, indexación y consulta de datos espaciales.

2.9 Metadatos para Tablas Espaciales

Oracle Spatial trata a todos los objetos en una única columna SDO_GEOMETRY de una tabla como una capa espacial.

Para realizar la validación, creación de índices, y consultas con respecto a cada capa espacial es necesario especificar los metadatos adecuados para cada capa. Esto incluirá la siguiente información:

- El número de dimensiones.
- Los límites de cada dimensión.
- La tolerancia de cada dimensión.
- El sistema de coordenadas.

Esta información para cada capa espacial se rellena en el USER_SDO_GEOM_METADATA.

2.9.1 Metadatos Espaciales

Oracle Spatial proporciona el USER_SDO_GEOM_METADATA para almacenar los metadatos para las capas espaciales. Esta vista de metadatos tiene la estructura que se muestra en el Ejemplo 10.

```
SQL> DESCRIBE USER_SDO_GEOM_METADATA;
Name                                      Null?                    Type
----------------------------------------- -------------------------------- -------------------------
TABLE_NAME                                NOT NULL                 VARCHAR2(32)
COLUMN_NAME                               NOT NULL                 VARCHAR2(1024)
DIMINFO                                                            MDSYS.SDO_DIM_ARRAY
SRID                                                               NUMBER
```

Ejemplo 10. La vista USER_SDO_GEOM_METADATA

Las columnas TABLE_NAME y COLUMN_NAME, identifican exclusivamente cada capa espacial. Además, para cada capa, el metadato almacena información sobre las dimensiones de cada capa en el atributo DIMINFO, y la información sobre el sistema de coordenadas de los datos geométricos en el atributo SRID.

Atributo SRID

Este atributo especifica el sistema de coordenadas en el que se ubican los datos de la capa espacial. El sistema de coordenadas podría ser uno de los siguientes:

- Geodésico: Coordenadas angulares, expresadas en términos de longitud, latitud con respecto a la superficie de la Tierra.
- Proyección: Coordenadas cartesianas que resultan de la realización de una transformación matemática de una zona de la superficie de la Tierra a un plano.
- Local: Sistemas de coordenadas cartesianos, sin ningún vínculo en la superficie de la Tierra y que se usan en ocasiones específicas para una aplicación determinada. Éstos se utilizan en CAD/CAM y otras aplicaciones donde los datos espaciales no corresponden a lugares en la Tierra.

Los diferentes sistemas de coordenadas geodésicos y proyectados están concebidos para maximizar la exactitud (en las distancias y en otros cálculos de relación espacial) para las diferentes partes o regiones del mundo.

En el caso de los sistemas de coordenadas geodésicos, se puede consultar la tabla de posibles valores MDSYS.CA_SRS mediante la selección de las filas en la columna WKTEXT que comienza con un prefijo de ‘GEOGCS’.

Atributo DIMINFO

Los datos espaciales son multidimensionales. El atributo DIMINFO en USER_SDO_GEOM_METADATA especifica la información sobre cada dimensión de la capa especificada. Dicho atributo es de tipo MDSYS.SDO_DIM_ARRAY. En el Ejemplo 11 se muestra la estructura.

```
SQL> DESCRIBE SDO_DIM_ARRAY;
 SDO_DIM_ARRAY VARRAY (4) OF MDSYS.SDO_DIM_ELEMENT
Name                                      Null?    Type
----------------------------------------- -------- ------------------------
SDO_DIMNAME                                        VARCHAR2(64)
SDO_LB                                             NUMBER
SDO_UB                                             NUMBER
SDO_TOLERANCE                                      NUMBER
```

Ejemplo 11. Estructura de SDO_DIM_ARRAY.

Hay que tener en cuenta que SDO_DIM_ARRAY es un array de longitud variable (VARRAY) de tipo SDO_DIM_ELEMENT. Cada SDO_DIM_ARRAY tiene un tamaño determinado por el número de dimensiones (así para una geometría de dos dimensiones, el atributo DIMINFO contendrá dos tipos de SDO_DIM_ELEMENT, y así sucesivamente).

Cada tipo SDO_DIM_ELEMENT almacena información para una dimensión específica y consta de los siguientes campos:

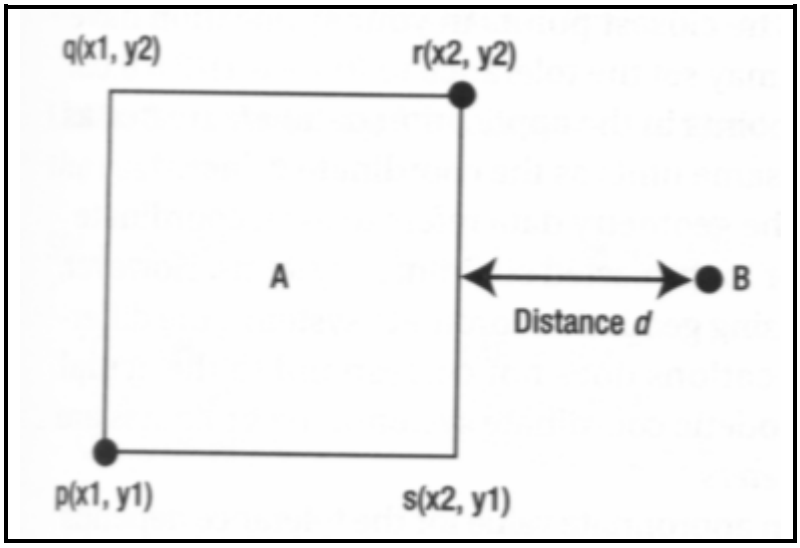
- SDO_DIMNAME: Este campo almacena el nombre de la dimensión. Por ejemplo, se puede llamar 'Longitud' o 'Latitud' para indicar que la dimensión representa la longitud o la latitud. El nombre que se especifica aquí no es interpretado por Oracle Spatial. Se puede usar 'X' para la longitud e 'Y' para la latitud.
- SDO_LB y SDO_UB: Estos dos números definen el límite inferior y el límite superior para los valores de una dimensión específica. Por ejemplo, el rango de valores para la dimensión de longitud es de -180 a 180, de modo que se asignaría -180 a SDO_LB y 180 a SDO_UN. Asimismo, para la latitud, SDO_LB y SDO_UB se establecerán a -90 y 90 respectivamente. Hay que tener en cuenta que estos límites son para una aplicación específica.
- SDO_TOLERANCE: Un valor SDO_TOLERANCE, o simplemente un valor de tolerancia, se utiliza para especificar un grado de precisión para los datos espaciales. Hay que especificar la distancia a la que dos valores deben estar separados para ser considerados diferentes. Por ejemplo, si la tolerancia se fija a 0,5, y la distancia entre dos puntos A y B es inferior a 0,5, entonces los puntos A y B se consideran en el mismo lugar.

  Por defecto, el valor de la tolerancia está en las mismas unidades que los valores de SDO_LB y SDO_UB. Sin embargo, en los sistemas de coordenadas geodésicos, el valor de la tolerancia es siempre en metros (mientras que los límites de SDO_LB y SDO_UB están en grados). Oracle, además, exige que la tolerancia tenga el mismo valor en todas las dimensiones.

Comprender la Tolerancia

Como se señaló la tolerancia se especifica como un campo del atributo DIMINFO en la vista USER_SDO_GEOM_METADATA. Los índices espaciales y otras operaciones de capas espaciales usan el atributo DIMINFO y la tolerancia asociada a ésta vista.

La configuración de valores incorrectos de la tolerancia puede producir resultados incorrectos e inesperados en múltiples funciones. A continuación se explica con un ejemplo:



Ejemplo 12. La tolerancia y su impacto sobre la validez y la relación de dos objetos, A y B.

El Ejemplo 12 muestra dos objetos, A y B. El objeto A es un rectángulo con cuatro vértices: p, q, r, y s. El vértice inferior izquierdo p, tiene las coordenadas (x1, y1), y el vértice superior derecho, tiene las coordenadas (x2, y2). La distancia entre los objetos A y B es d. La relación espacial entre los objetos A y B, y la geometría del objeto A es considerada válida o no valida, variara dependiendo de cómo se establezca el valor de la tolerancia.

- Relación entre A y B: Si la distancia d < tolerancia, entonces B se considera el limite exterior de A. En otras palabras, el objeto A es considerado como la intersección con B.

  Si la distancia d >= tolerancia, entonces A y B son considerados disjuntos o, en otras palabras, no intersección.
- Comprobación de la validación del objeto A: Si la distancia entre p y s es menor que el valor de la tolerancia – es decir, (x2 – x1) < tolerancia – entonces p y s se consideran puntos/vértices duplicados. Lo mismo ocurrirá con q y r. Oracle Spatial no permite puntos duplicados en la especificación de una geometría, así la geometría del objeto A debería ser considerada inválida. Lo mismo ocurre si la distancia (y2 – y1) entre p y q o r y s es menor que la tolerancia.

  Si las distancias anteriores son iguales o superiores a la tolerancia, entonces los vértices son considerados distintos y la geometría del objeto A es considerada una geometría válida de Oracle Spatial.

De este ejemplo, se desprende que la tolerancia juega un papel importante en la aplicación, por lo que es necesario ajustarla adecuadamente.

Elegir el Valor de la Tolerancia

Como norma general, el valor de la tolerancia se debe establecer en la menor distancia distinguible en la aplicación. En la mayoría de los casos, esta distancia corresponde a la mitad de la diferencia entre los valores de dos coordenadas. Por ejemplo, si los puntos más cercanos tienen los valores de 0,1 y 0,2 en una dimensión especifica, se puede configurar la tolerancia a (0,2 – 0,1) / 2 = 0,05. Esto asegurará que los dos puntos (y todos los otros puntos de la aplicación) sean tratados como distintos.

Esta técnica puede ser aplicada directamente cuando los datos geométricos se refieren a sistemas de coordenadas locales (como en *CAD/CAM* y otras aplicaciones), o para sistemas de coordenadas proyectadas. Sin embargo, para las localizaciones en la superficie de la Tierra utilizando el sistema de coordenadas geodésico, la diferencia del valor de la longitud o la latitud de dos localizaciones no se corresponde con la distancia real que hay entre ellos. En estos casos, las coordenadas se interpretan en grados y la tolerancia en metros.

Está claro, que la especificación de un valor adecuado para la tolerancia depende del sistema de coordenadas (es decir, del atributo SRID). En la Tabla 1, se describen algunas recomendaciones para los diferentes sistemas de coordenadas.

Sistema de Coordenadas	Valores SRID	Tolerancia	Unidades
Sistema de coordenadas Geodésico	Select SRID from MDSYS.CS_SRS, where WKTEXT is like 'GEOGCS%' (por ejemplo, 8265, 8307).	0.5 (no debe ser menor que 0.05)	Metros para la tolerancia; grados para la longitud y latitud.
Sistema de coordenadas Proyectado	Select SRID from MDSYS.CS_SRS, where WKTEXT is like 'PROJCS%' (por ejemplo, 32774).	La mitad de la diferencia más pequeña entre dos valores en una dimensión.	Las unidades para la tolerancia son las mismas que las unidades para las coordenadas en la dimensión.
Sistema de coordenadas Local	Select SRID from MDSYS.CS_SRS, where WKTEXT is like 'LOCAL_CS%'.	La mitad de la diferencia más pequeña entre dos valores en una dimensión.	Las unidades para la tolerancia son las mismas que las unidades para las coordenadas en la dimensión.
Sistema de coordenadas no específico	NULL	La mitad de la diferencia más pequeña entre dos valores en una dimensión.	Las unidades para la tolerancia son las mismas que las unidades para las coordenadas en la dimensión.

Tabla 1. Sugiere Valores para la Tolerancia Basados en el SRID para las Aplicaciones

2.10 El tipo de datos SDO_GEOMETRY

A continuación, se habla del almacenamiento y modelado de los diferentes tipos de información de localización usando el tipo de dato SDO_GEOMETRY en Oracle. El tipo SDO_GEOMETRY puede almacenar una gran variedad de datos espaciales, incluyendo los siguientes:

- Un punto, que pude ser usado para almacenar las coordenadas de ubicación, por ejemplo, de un cliente, una tienda, una dirección de entrega, etc.
- Una cadena de línea, que puede ser usada para almacenar la ubicación y la forma de un segmento de la carretera.

- Un polígono, que puede ser usado para almacenar los límites de una ciudad, empresas, regiones, etc.
- Geometrías complejas, tales como múltiples polígonos, que puede ser usado para almacenar las fronteras de estados como Texas, Hawai o California.

Primero, se tiene una visión detallada de la estructura del SDO_GEOMETRY: los diferentes atributos y los valores que puede tomar para almacenar los diferentes tipos de datos geométricos.

Después, se estudia como hay que construir realmente los objetos SDO_GEOMETRY para geometrías simples, tales como puntos, líneas y polígonos.

Por último, se muestra como construir geometrías más complejas, tales como multipolígonos.

2.10.1 Tipos de Geometrías Espaciales en Oracle

Veamos que tipos de datos espaciales puede almacenar el SDO_GEOMETRY. La Imagen 7 ilustra algunos de estos tipos.

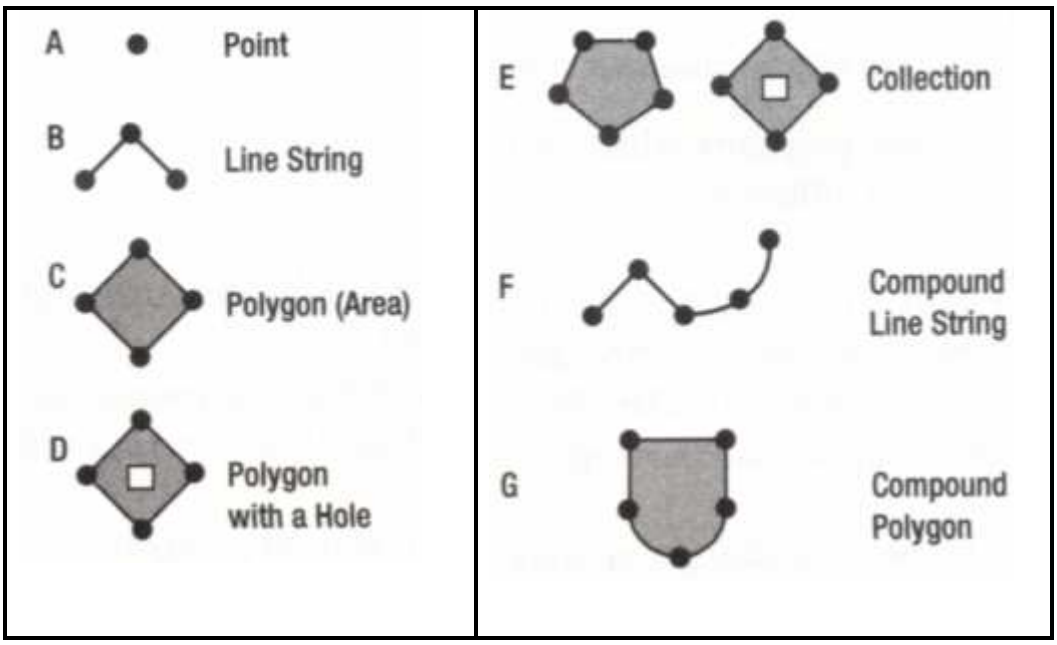


Imagen 7. Ejemplos de datos espaciales que SDO_GEOMETRY puede representar

Puntos

La geometría más simple es un punto. Un punto puede representar la ubicación de un cliente, un sitio de entrega, o una tienda. El objeto A en la 7 es un ejemplo de una geometría de punto.

Cadenas de Línea

Una cadena de línea conecta varios puntos (o vértices como hacen referencia algunas veces). En general, las carreteras, las redes de transporte, las líneas de servicios públicos, y las tuberías son representadas como cadenas de línea de tipo SDO_GEOMETRY. Si la cadena de línea es cerrada, entonces es un anillo. De lo contrario, es sólo una línea. Una cadena de línea conecta dos o más puntos por:

- Líneas rectas: se hace referencia a esto como una simple cadena de línea recta o como una cadena de línea cuando no hay ambigüedad. El objeto B en la Imagen 7 es un ejemplo de cadena de línea recta.
- Arcos circulares: se hace referencia a esto como una cadena de arco.
- Combinación de líneas rectas y arcos circulares: se hace referencia a esto como una composición de cadena de línea (curva). El objeto F en la 7 es un ejemplo de una composición de cadena de línea.

Polígonos

Un polígono está especificado por uno o más anillos (cadenas de línea cerradas) y es asociado con un área. El objeto C de la Imagen 7 es un área poligonal delimitada por líneas rectas que unen cuatro puntos (área sombreada que se muestra en la Imagen 7). En este ejemplo, el objeto C tiene forma de diamante, pero en general un polígono puede tener cualquier forma arbitraria. Un polígono puede representar un límite de una ciudad, una zona buffer de un sitio de almacenamiento. Un polígono tiene las siguientes propiedades:

- El perímetro de un polígono se compone de uno o más anillos (una cadena de línea cerrada). Hay casos especiales de polígonos que pueden ser especificados fácilmente en SDO_GEOMETRY incluidos rectángulos y círculos.
- Un polígono, a diferencia de una cadena de línea, se asocia con una zona delimitada por la frontera. El área tiene que ser contigua – es decir, los bordes del polígono no se pueden cruzar. Esto significa que el dígito 8 no puede ser un polígono válido. (Sin embargo, el dígito 8 puede ser modelado como un multipolígono o colección de geometría). El objeto C de la Imagen 7 es un ejemplo de un polígono (válido).
- El anillo especifica el perímetro o colección de un polígono que puede estar compuesto por líneas rectas, arcos, o una combinación de arcos y líneas. Si es una combinación de arcos y líneas, se refiere al polígono como un polígono compuesto. El objeto G en la Imagen 7 es un ejemplo de un polígono compuesto, y cómo su perímetro está conectado por líneas rectas y arcos.
- El área cubierta por un polígono puede expresarse usando un anillo exterior y algún número de (cero o más) anillos internos. Los anillos internos son denominados agujeros o huecos vacíos, de manera que el área total del polígono será la zona abarcada por el anillo exterior menos la delimitada por sus agujeros. El objeto D en la Imagen 7 muestra un polígono con un anillo exterior y un anillo interior. El anillo interior en este ejemplo es un rectángulo. El área cubierta por este polígono es la región sombreada entre los dos anillos.

Colecciones

Una colección tiene múltiples elementos geométricos. Una colección puede ser heterogénea – es decir, podría ser cualquier combinación de puntos, líneas y polígonos. Alternativamente, una colección puede ser homogénea – es decir, podría consistir en elementos de un sólo tipo. Los tipos específicos de estas colecciones homogéneas son multipunto, multilínea, o colecciones de multipolígonos.

El objeto E en la Imagen 7 tiene dos polígonos, un polígono en forma de pentágono y un polígono con un vacío, y es un ejemplo de una colección multipolígono. Las regiones sombreadas de la figura muestran el área cubierta por esta geometría.

Aplicación lógica de SDO_GEOMETRY

En general, la forma de los objetos espaciales puede ser bastante compleja, requieren un gran número de puntos conectados (o vértices). Cualquier tipo de datos que modelen datos espaciales deben ser capaces de representar una gran variedad de formas – desde segmentos de carreteras complejos a una forma arbitraria de una ciudad y los límites de una propiedad.

Para representar estas formas geométricas complejas, el tipo SDO_GEOMETRY es implementado usando un array de elementos como se muestra en la Imagen 8.

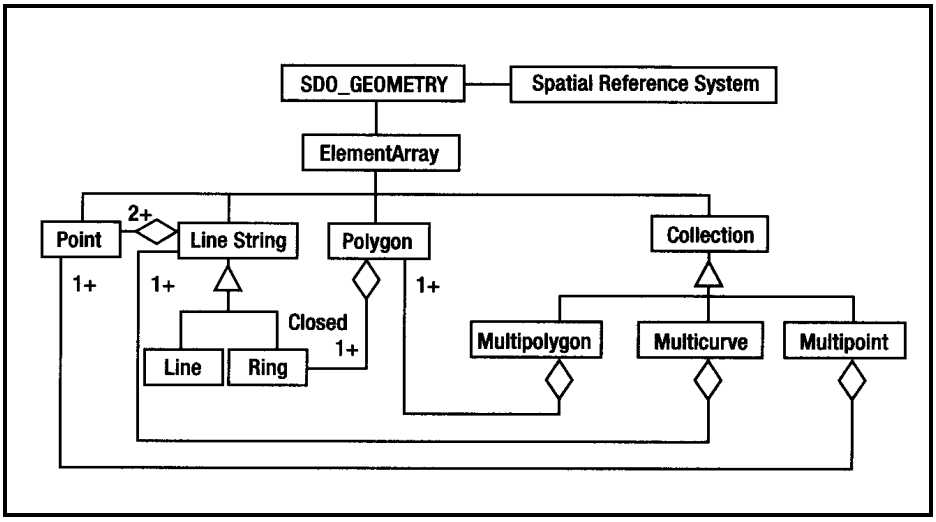


Imagen 8. Diagrama de clases conceptual del tipo de datos SDO_GEOMETRY.

El tipo de datos SDO_GEOMETRY tiene dos componentes lógicos: el sistema de referencia espacial (también llamado sistema de coordenadas) de la geometría y el ElementArray.

El ElementArray, o array de elementos, describe la forma y la ubicación del SDO_GEOMETRY (con referencia al sistema de coordenadas especificado). Este array de elementos constituye (o representa) el objeto SDO_GEOMETRY. El array de elementos representa cualquiera de los diferentes tipos de datos espaciales representados en la Imagen 7: punto, cadena de línea, polígono o colección. Esto se muestra en la Imagen 8 por la relación es-un, ilustrado por el símbolo del triángulo entre estos tipos y el ElementArray. Hay que tener en cuenta que el símbolo del rombo muestra una relación muchos-a-uno entre los diferentes tipos. Por ejemplo, un rombo entre un punto y una cadena de línea indica que “muchos” puntos constituyen “una” cadena de línea. Hay que tener en cuenta que “2+” al lado del rombo indica el número mínimo. Por ejemplo, al menos, dos o más puntos forman una cadena de línea. Asimismo, se observa que uno o más anillos constituyen un polígono, y que uno o más polígonos constituyen un multipolígono.

2.10.2 Tipo SDO_GEOMETRY, Atributos y Valores

Ahora que se sabe lo que es un SDO_GEOMETRY, lo que puede representar y cómo está constituido internamente, se va a examinar su estructura en Oracle. El Ejemplo 13 describe el tipo de datos SDO_GEOMETRY.

SQL> DESCRIBE **SDO_GEOMETRY**

Name	Null?	Type
SDO_GTYPE		NUMBER
SDO_SRID		NUMBER
SDO_POINT		SDO_POINT_TYPE
SDO_ELEM_INFO		SDO_ELEM_INFO_ARRAY
SDO_ORDINATES		SDO_ORDINATE_ARRAY

Ejemplo 13. Tipo de Dato SDO_GEOMETRY en Oracle.

El tipo de datos SDO_GEOMETRY está formado por los siguientes atributos:

- El atributo SDO_GTYPE especifica que tipo de forma (punto, línea, polígono, colección, multipunto, multilínea, o multipolígono) representa en realidad la geometría. Aunque éste atributo capta lo que el tipo de geometría está representando, no especifica las coordenadas reales.
- El atributo SDO_SRID especifica el identificador del sistema de referencia espacial (sistema de coordenadas) en el cual se especifica la forma de la geometría.

En el Ejemplo 13, se observa que la geometría se compone de un elemento de tipo array. Para especificar las coordenadas de los elementos, se puede hacer de una de las siguientes maneras:

- Si la geometría es un punto (por ejemplo, la ubicación de los clientes), entonces se puede almacenar las coordenadas en el atributo SDO_POINT del SDO_GEOMETRY.
- Si la geometría es una forma arbitraria (por ejemplo, una red de calles o los límites de una ciudad), entonces se puede almacenar las coordenadas utilizando los atributos de tipo array SDO_ORDINATES y SDO_ELEM_INFO.

  - El atributo SDO_ORDINATES almacena las coordenadas de todos los elementos de la geometría.
  - El atributo SDO_ELEM_INFO especifica en el array SDO_ORDINATES cuando comienza un nuevo elemento, la forma en la que está conectado (por líneas rectas o arcos), y si se trata de un punto (aunque se recomienda utilizar el SDO_POINT para el almacenamiento), una línea, o un polígono.

A continuación, se describen cada uno de estos atributos con más detalle.

Atributo SDO_GTYPE

Este atributo describe el tipo de forma geometría del objeto. Esto refleja aproximadamente los niveles más altos en la jerarquía de clases de la Imagen 8. En concreto, se tiene un valor distinto para indicar si la geometría es un punto, una línea, un polígono, un multipunto, un multipolígono, una multilínea o una colección arbitraria. La geometría del objeto en sí, puede ser una combinación de varios elementos, cada uno de una forma diferente. Sin embargo, este atributo especifica el tipo general para todo el objeto (con todos los elementos de los que se compone).

El atributo SDO_GTYPE es un número de cuatro dígitos con la siguiente estructura: D00T. El primero y el último dígito toman valores diferentes basados en la dimensionalidad y en la forma geométrica, como se describe en la Tabla 2. El segundo y el tercer dígito se establecen a 0.

Dígito	Valores
D (dimensión de la geometría)	2 = Dos dimensiones 3 = Tres dimensiones 4 = Cuatro dimensiones
T (forma / tipo de la geometría)	0 = Tipo No Interpretado 1 = Punto 5 = Multipunto 2 = línea 6 = Multilínea 3 = polígono 7 = Multipoligono 4 = Colección

Tabla 2. Los valores para D, T en Formato D00T del atributo SDO_GTYPE de SDO_GEOMETRY.

El dígito D en la representación D00T del atributo SDO_GTYPE se utiliza para almacenar la dimensionalidad de la geometría del objeto. Spatial puede trabajar con dos a cuatro dimensiones geométricas. Si la geometría es bidimensional, entonces cada vértice tiene dos coordenadas en la forma geométrica. Si la geometría es tridimensional, cada vértice tiene tres coordenadas, y así sucesivamente. Estas coordenadas de los vértices se almacenan en el atributo SDO_ORDINATES (o SDO_POINT) que se explicará más adelante.

El dígito T en el atributo SDO_GTYPE especifica el tipo/forma de la geometría. Para los tipos simples, tales como puntos, líneas y polígonos, T está entre 1 y 3. Para colecciones, el valor de T del tipo simple siempre se le sumará 4. Por ejemplo, T para un punto es 1, y para un multipunto es 1 + 4 = 5. Del mismo modo, T para una línea es 2, y para una multilínea es 2 + 4 = 6, y así sucesivamente.

El valor de T (para el atributo SDO_GTYPE) es 1 si la geometría se compone de un único punto, y 5, si la geometría tiene varios puntos. Por ejemplo para un objeto de la Imagen 8, el valor de T es 1 y el valor del atributo SDO_GTYPE es 2001.

El valor de T es 2 si la geometría representa una línea. Esta línea puede ser una línea simple conectada por un número cualquiera de puntos, por líneas rectas o por arcos. Alternativamente, esta línea puede ser una combinación de múltiples partes especificando segmentos de línea recta y arcos. Hay que tener en cuenta que la línea sigue siendo continua. Si la geometría está formada por segmentos de varias líneas que no están conectados, entonces el tipo es 6 (multilínea). Para los objetos B y F de la Imagen 7, el valor de T es 2 y el SDO_GTYPE es 2002.

El tipo T es 3 si la geometría representa un área limitada por una línea cerrada (también conocida como anillo). Los límites pueden ser conectados por líneas, arcos, o una combinación de ambos. El polígono puede contener uno o más anillos interiores llamados huecos. En tales casos, el área del polígono se calcula substrayendo las áreas de los huecos. El área cubierta por una geometría que tiene T igual a 3 debe ser contigua. Los objetos C, D y G son algunos ejemplos. Hay que tener en cuenta que el objeto D tiene una zona exterior y un anillo interior (rectángulo), pero todavía hay una única área “contigua” mostrada por la región sombreada. Por lo tanto, se considera un único polígono con el tipo T establecido a 3.

Si hay más de un polígono (no vacío) en la geometría (es decir, si el área de la geometría no es contigua), entonces es una geometría multipolígono y el tipo es 7. El objeto E de la Imagen 8 es un ejemplo de esto.

Si la geometría es una colección de puntos, líneas y/o polígonos, la geometría es una colección de geometría. El valor de T para esta geometría es 4. El objeto E, cuenta con dos polígonos, se puede establecer el tipo a 7, un multipolígono. Alternativamente, se puede establecer la descripción más genérica de una colección. El tipo T en este caso será 4.

Hay que tener en cuenta que la mayor de las figuras representa sólo geometrías en dos dimensiones. En casi todos lo casos, Oracle Spatial no interpreta los valores de la tercera y cuarta dimensión, incluso si los valores se especifican. No obstante, si se quieren usar tres (o cuatro) dimensiones sólo hay que configurar el dígito D en D00T para el atributo SDO_GTYPE a 3 (ó 4). Oracle Spatial, permite almacenar tres (o cuatro) coordenadas para cada vértice de la geometría, así que, si se tienen dos puntos (1,1,4) y (2,2,5), y se especifica el atributo SDO_GTYPE a 3002 (una línea), a continuación, Oracle Spatial trata a la geometría como una línea que une los puntos (1,1) y (2,2). Esto permite utilizar la tercera y cuarta dimensión para contener la información adicional que puede ser almacenada con cada vértice de la geometría. Por ejemplo, la tercera dimensión podría modelar la altura de cada vértice en la geometría.

Así, Oracle Spatial no interpreta la tercera y cuarta dimensión de forma predeterminada. Oracle Spatial proporciona algunas funciones para operar en estas dimensiones para aplicaciones específicas. Además, Oracle Spatial puede apoyar el procesamiento explícito de objetos específicos de tres y cuatro dimensiones. Estos incluyen los siguientes tipos de geometrías:

- Geometría de punto, multipunto: Para una geometría de un punto de tres dimensiones, el tipo T es 1 para un punto y 5 para un multipunto.
- Polígono cúbico, polígonos multicúbicos: El tipo T es de nuevo 3 para un cubo individual y 7 para múltiples.

Atributo SDO_SRID

Este atributo especifica el sistema de referencia espacial, o el sistema de coordenadas, para la geometría. Para entender mejor lo que es un sistema de coordenadas, se considera el ejemplo de la Imagen 9.

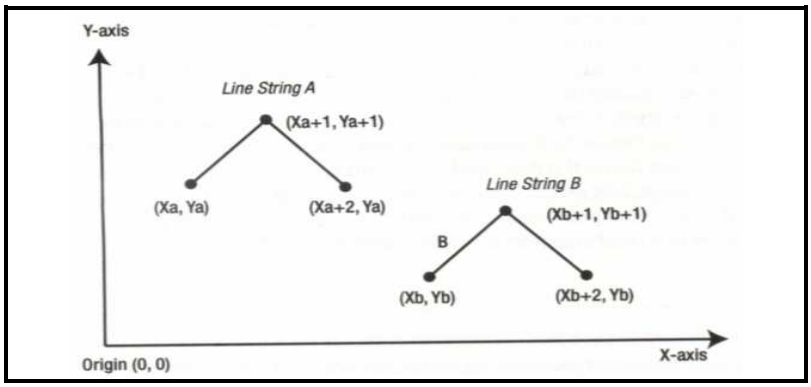


Imagen 9. Ejemplo de Sistemas de Coordenadas

Hay que tener en cuenta que en la Imagen 9, las ubicaciones de los dos objetos A y B, se especifican con respecto al origen y usando las coordenadas de los ejes X e Y. Además, A y B tienen la misma forma, sin embargo, su posición con respecto al origen es diferente. Si se cambia el origen, las coordenadas de los dos objetos también cambian. Éste marco de referencia usando los ejes X e Y se llama Sistema Cartesiano.

Para saber si esta representación es buena, se examina la superficie de la Tierra. La Tierra tiene aproximadamente forma elipsoidal. La ubicación ha sido tradicionalmente especificada usando las líneas de longitud y latitud en la Tierra. El aplanamiento de la superficie de la Tierra a un plano de dos dimensiones pierde la proximidad espacial y distorsiona la forma.

Existen dos técnicas generales para las representaciones de las ubicaciones en la superficie de la Tierra sin inexactitudes ni distorsiones. La primera es modelar la Tierra usando una superficie elipsoidal en tres dimensiones. La segunda es proyectar los datos en un plano de dos dimensiones.

Sistema de Coordenadas Geodésicas

Si se modela la superficie de la Tierra como un elipsoide regular de tres dimensiones, se puede medir la distancia entre los objetos mediante el cálculo de las distancias de los lugares correspondientes en el elipsoide. Desafortunadamente, la Tierra no es un elipsoide perfecto, y por tanto no se puede medir con precisión todas las áreas de la Tierra. Esto llevó a los geógrafos a definir múltiples elipsoides para adaptarse a sus necesidades. Oracle Spatial facilita referencias elipsoidales en la tabla MDSYS.SDO_ELLIPSOIDS.

A veces en el modelado, se tendrá que cambiar el centro de la Tierra y rotas los ejes para adaptarlos mejor a la curvatura de la región local. Para ello, se pueden crear modelos denominados datums desplazando y rotando elipsoides específicos para adaptar mejor la curvatura de la Tierra en las distintas regiones. Se pueden ver los distintos modelos tridimensionales del datums en la tabla MDSYS.SDO_DATUMS. Posicionar las coordenadas (longitud y latitud) de un dato concreto en la superficie de la Tierra se conoce como un sistema de coordenadas geodésicas (o sistema geodésico de referencia espacial).

Sistema de Coordenadas Proyectadas

En la mayoría de las aplicaciones, los datos se concentran en una pequeña región de la Tierra. Proyectando estos datos en plano de dos dimensiones se obtiene una representación más simple y más precisa para la aplicación. Para proyectar los datos de la superficie de la Tierra a una superficie plana de dos dimensiones, primero hay que comenzar con un modelo (datum) de tres dimensiones de la Tierra. Después, se usa una variedad de técnicas de proyección, los dato tridimensionales son transformados en datos bidimensionales sobre una superficie plana.

Hay diferentes técnicas de proyección porque no hay una técnica única que pueda proyectar la forma tridimensional a bidimensional preservando las distancias entre objetos, las áreas de objetos de gran tamaño, las direcciones, etc.

Resumiendo, para la elección de una proyección y un datum de tres dimensiones, las ubicaciones en la superficie de la Tierra pueden ser representadas en un plano de dos dimensiones. Tales referencias utilizan datum específicos y una proyección adecuada se refiere como un sistema de coordenadas proyectado o un sistema proyectado de referencia espacial.

Elegir un Sistema de Coordenadas apropiado

Se elige el sistema de coordenadas mediante el establecimiento de un valor apropiado para el atributo SDO_SRID. A continuación, se describe cómo determinar los valores apropiados para proyectados, geodésicos y sistemas de coordenadas locales.

Si la geometría no hace referencia a una ubicación sobre la superficie de la Tierra, pero se refiere al diseño en *CAD/CAM* u otras aplicaciones, entonces hay que ponerlo a NULL o darle un valor determinado por el proveedor de los datos.

De lo contrario, si la geometría se refiere a una ubicación sobre la superficie de la Tierra, el atributo SDO_SRID toma un valor que corresponde a un sistema de coordenadas de proyección o a un sistema de coordenadas geodésicas. El sistema de coordenadas de proyección se usa cuando todos los datos se encuentran en una pequeña región de la Tierra. Los sistemas de coordenadas de proyección son útiles para satisfacer las necesidades de las aplicaciones como la preservación de las distancias entre los lugares, las formas, o las áreas de la geometría de los objetos. Los sistemas de coordenadas

geodésicas son útiles si los datos se encuentran en un porcentaje mucho mayor en la superficie de la Tierra, y los pequeños errores en algunas propiedades geométricas como distancias, áreas, se pueden tolerar. Se pueden buscar los SDO_SRIDs para los sistemas de coordenadas geodésicos o proyectados en la tabla MDSYS.CS_SRS. El Ejemplo 14 muestra las columnas de esta tabla.

SQL> DESCRIBE **MDSYS.CS_SRS**

Name	Null?	Type
CS_NAME		VARCHAR2(68)
SRID	NOT NULL	NUMBER(38)
AUTH_SRID		NUMBER(38)
AUTH_NAME		VARCHAR2(256)
WKTEXT		VARCHAR2(2046)
CS_BOUNDS		SDO_GEOMETRY

Ejemplo 14. Tabla MDSYS.CS_SRS

La tabla MDSYS.CS_SRS tiene las siguientes columnas:

- CS_NAME: Especifica el nombre de los sistemas de coordenadas.
- SRID: Es el identificador del sistema de referencia espacial. Este identificador es único para cada referencia espacial o sistema de coordenadas.
- AUTH_SRID y AUTH_NAME: Valores asignados al sistema de coordenadas.
- WKTEXT: Proporciona una descripción detallada del sistema de coordenadas. Para los sistemas de coordenadas geodésicos, este campo comienza con el prefijo 'GEOGCS' y para los sistemas de coordenadas proyectados, con 'PROJCS'.
- CS_BOUNDS: Especifica una geometría donde el sistema de coordenadas es válido. El almacenamiento de datos más allá de los límites puede llevar a resultados inexactos. Normalmente este atributo es NULL.

Atributo SDO_POINT

Este atributo se utiliza para especificar la ubicación de un punto de la geometría, como por ejemplo la ubicación de un cliente. Este atributo es de tipo SDO_POINT_TYPE, que es otro tipo de objeto. El Ejemplo 15 muestra la estructura de este tipo.

SQL> DESCRIBE **SDO_POINT_TYPE**

Name	Null?	Type
X		NUMBER
Y		NUMBER
Z		NUMBER

Ejemplo 15. Tipo de Dato SDO_POINT_TYPE

El SDO_GTYPE para un punto se establece a D001. El punto A en la Imagen 10, identificado por las coordenadas Xa e Ya representa la ubicación de un cliente.

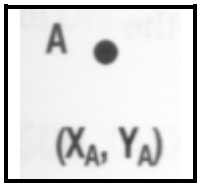


Imagen 10. Ejemplo de un punto con coordenadas Xa, Ya

El Ejemplo 16 muestra como insertar el objeto SDO_GEOMETRY en la tabla geometry_examples para representar el punto A (sustituyendo (-79, 37) que son las coordenadas reales).

```
SQL> INSERT INTO geometry_examples (name, description, geom.) VALUES
(
  'POINT',
  '2-dimensional Point at coordinates (-79, 37) with srid set to 8307',
  SDO_GEOMETRY
  (
    2001,       -- SDO_GTYPE formato: D00T. Se establece a 2001 para un punto de dos dimensiones
    8307,       -- SDO_SRID (geodetic)
    SDO_POINT_TYPE
    (
      -79,      -- valor para la Longitud
       37,      -- valor para la Latitud
      NULL      -- no hay tercera dimension (solo 2 dimensiones)
    ),
    NULL,
    NULL
  )
);
```

Ejemplo 16. Datos de un Punto en la tabla geometry_examples

Los objetos en Oracle son instanciados usando el constructor del objeto correspondiente. La columna 'geom' es un objeto de tipo SDO_GEOMETRY y es instanciada como se muestra en el Ejemplo 15. Los campos de este objeto se rellenan de la siguiente manera:

- SDO_GTYPE: El formato es D00T, don D es 2 y T es 1 para un punto de dos dimensiones.
- SDO_SRID: Se establece a 8307.
- SDO_POINT: Establece las coordenadas x e y en SDO_POINT_TYPE a (-79, 37) en el Ejemplo 15. La coordenada z se establece a NULL.
- SDO_ELEM_INFO: No se utiliza; se establece a NULL.
- SDO_ORDINATES: No se utiliza; se establece a NULL.

Hay que tener en cuenta que el SDO_POINT puede almacenar sólo tres coordenadas (x, y, z). Ésta representación es adecuada si los datos tienen tres o menos dimensiones. Para cuatro dimensiones, hay que utilizar los atributos SDO_ELEM_INFO y SDO_ORDINATES.

Atributos SDO_ELEM_INFO y SDO_ORDINATES

Es posible que se desee almacenar elementos más complejos que los puntos, como líneas y polígonos, necesitando para ello un gran número de vértices. Para almacenar elementos tan complejos, se van a utilizar otras dos estructuras del tipo SDO_GEOMETRY, los atributos SDO_ORDINATES y SDO_ELEM_INFO. En conjunto, estos atributos permiten especificar los diferentes elementos que componen una geometría: el SDO_ORDINATES almacena las coordenadas de los vértices de todos los elementos de una geometría, y el SDO_ELEM_INFO específica el tipo de elementos y donde comienzan.

Primero, hay que entender cómo representar los elementos utilizando los atributos SDO_ELEM_INFO y SDO_ORDINATES.

Atributo SDO_ORDINATES

Este atributo almacena las ordenadas en todas las dimensiones de todos los elementos de una geometría. El atributo SDO_ORDINATES es de tipo SDO_ORDINATE_ARRAY, que es una colección de tipo VARRAY (array de longitud variable) de números. El VARRAY es útil para almacenar los puntos que describen una forma geométrica en el orden adecuado. Si la dimensionalidad del dato es D, entonces cada numero consecutivo D en el SDO_ORDINATES especifica las coordenadas de un vértice. Por ejemplo, si se quiere modelar una línea que une el punto A que tiene las coordenadas (Xa, Ya) con el punto B que tiene las coordenadas (Xb, Yb), entonces el SDO_ORDINATES contendrá los número Xa, Ya, Xb, Yb y, en ese orden. El tamaño de este atributo es de 1048576. Este límite de gran tamaño proporciona suficiente espacio para almacenar los vértices de una gran geometría compleja.

Si el atributo SDO_ORDINATES especifica las ordenadas (en todas las dimensiones) de todos los elementos de la geometría de un objeto, para representar los diferentes elementos que componen dicha geometría se necesita interpretar y separar las ordenadas de los elementos especificando ésta información en el atributo SDO_ELEM_INFO.

Atributo SDO_ELEM_INFO

El atributo SDO_ELEM_INFO es de tipo SDO_ELEM_INFO_ARRAY, que es también un VARRAY de números con un tamaño máximo de 1.048.576 números. Cada tres números consecutivos, en el SDO_ELEM_INFO son agrupados en un triplete descriptor, que describe un elemento o una parte de un elemento. Así, lógicamente, el atributo SDO_ELEM_INFO es un array de tripletes (tres números). Esto significa que el tamaño de este array es siempre múltiplo de 3.

Cada triplete descriptor esta asociado con un elemento de la geometría. El triplete tiene la forma <offset, element-type, interpretation>. El offset especifica el comienzo del índice en el array SDO_ORDINATES donde están almacenadas las ordenadas de los elementos. Los ostros dos números, element-type e interpretation, toman diferentes valores dependiendo de si el elemento asociado representa un punto, una línea, o un polígono, y si los límites están conectados por líneas rectas, arcos o ambos.

Primero hay que buscar los valores del SDO_ELEM_INFO para los datos de las aplicaciones. Por ejemplo, en aplicaciones de negocio, las geometrías que se pueden construir son:

- Puntos que representan la ubicación de clientes, competidores, etc.
- líneas que representan calles y carreteras.
- Polígonos que representan los límites de las ciudades.

En la mayoría de los casos, estas geometrías tienen como máximo un triplete descriptor y representa a lo sumo un elemento de tipo punto, línea o polígono. Estos elementos y geometrías son elementos simples y geometrías simples. En estos casos, el triplete descriptor tiene los siguientes valores:

- Offset: Es siempre 1, ya que tiene solamente un elemento en el campo SDO_ORDINATES.
- Element-type; Tiene una correspondencia directa con el valor del tipo T en el SDO_GTYPE para la geometría.

    - Para puntos, el element-type es 1 (el valor de T en SDO_GTYPE es 1)
    - Para líneas, el element-type es 2 (el valor de T en SDO_GTYPE es también 2).
    - Para polígonos, el element-type es 3 (el valor de T en SDO_GTYPE es 3).

- Interpretation: Ésta en la única información que contiene un elemento.

    - Para un punto, interpretation es1.
    - Para líneas y polígonos, interpretation es 1 si la conectividad es por líneas rectas, y interpretation es 2 si la conectividad es por arcos. Por ejemplo, una línea conectada por líneas rectas tiene el SDO_ELEM_INFO A (1, 2, 1) (es decir, el offset a 1, el element-type a 2, y la interpretation a 1).
    - Para polígonos, la interpretation puede ser 3 indicando así, que el polígono es un rectángulo.
    - Asimismo, para polígonos con la interpretation a 4 indica que el polígono es un círculo.

La Tabla 3 resume los posibles valores para el array SDO_ELEM_INFO (y el array SDO_ORDINATES) basado sobre el tipo de elemento. Con estos valores, se puede construir un SDO_GEOMETRY además de dar los valores adecuados al los atributos SDO_GTYPE y SDO_SRID.

A continuación se detallan ejemplos de geometrías simples. Después, se describen geometrías más complejas con más de un triplete descriptor. Ejemplos de estos seria una calle formada por líneas rectas y arcos. Estas geometrías se denominan geometrías complejas.

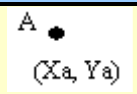
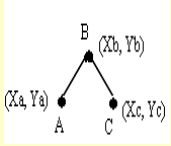
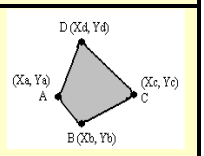
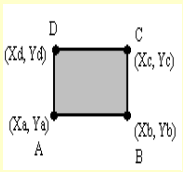
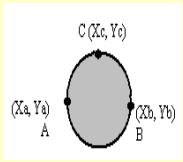
Nombre	Tipo Elemento (Etype)	Interpretación	SDO_ELEM_INFO: (1, Etype, Interpretación)	SDO_ORDINATES	Ilustración
Punto (Ej. Ubicación del cliente)	1	N, donde N es el número de puntos. 1 es para un punto simple.	(1, 1, 1)	(Xa, Ya)	
Cadena de línea (Ej. Calles, carreteras)	2	1 = Conectado por líneas rectas 2 = Conectado por arcos	(1, 2, 1) (1, 2, 2)	(Xa, Ya, Xb, Yb, Xc, Yc) (Xa, Ya, Xb, Yb, Xc, Yc)	
Polígono (Ej. Límites de una ciudad, zona de amortigua miento)	1003	1 = Límite del polígono conectado por líneas rectas. 3 = Polígono rectángulo (solo especifica las esquinas inferior izquierda y superior derecha) 4 = Polígono círculo(especifica tres puntos en el límite del círculo)	(1, 1003, 1) (1, 1003, 3) (1, 1003, 4)	(Xa, Ya, Xb, Yb, Xc, Yc, Xd, Yd, Xa, Ya) (Xa, Ya, Xc, Yc) (Xa, Ya, Xb, Yb, Xc, Yc)	  

Tabla 3. Valores para SDO_ELEM_INFO (y SDO_ORDINATES) para Geometrías Simples

2.10.3 Ejemplos de Geometrías Simples

Una geometría simple se compone de un solo triplete descriptor y representa un punto, una línea o un polígono. Las ordenadas de la geometría siempre se almacenan comenzando por el offset a 1 (ya que es sólo un elemento). Esto significa que el SDO_ELEM_INFO tiene siempre la forma (1, x, y).

Punto

En el Ejemplo 16, se vió como representar un punto de dos dimensiones usando el atributo SDO_POINT del SDO_GEOMETRY. Un mecanismo alternativo (pero no recomendado) es almacenar las coordenadas del punto en el array SDO_ORDINATES. El Ejemplo 17 muestra este almacenamiento.

```
SQL> INSERT INTO geometry_examples VALUES
(
  '2-D POINT stored in SDO_ORDINATES',
  '2-dimensional Point at coordinates (-79, 37) with srid set to 8307',
  SDO_GEOMETRY
  (
    2001,       -- SDO_GTYPE formato: D00T. Se establece a 2001 como para un punto de 2
dimensiones.
    8307,       -- SDO_SRID
    NULL,      -- SDO_POINT se establece a NULL
    SDO_ELEM_INFO_ARRAY       -- SDO_ELEM_INFO  (ver lo valores en la Tabla 3 )
    (
      1,       -- Offset es 1
      1,       -- Element-type es 1 para un punto
      1        -- Interpretación especifica # para puntos. En este caso 1.
    ),
    SDO_ORDINATE_ARRAY       -- SDO_ORDINATES
    (
      -79,       -- Valor para la Longitud
       37,       -- Valor para la Latitud
     )
  )
);
```

Ejemplo 17. Almacenamiento de las coordenadas de un punto en el array SDO_ORDINATES

En el Ejemplo 17, hay que destacar que el objeto SDO_GEOMETRY es instanciado utilizando el objeto constructor con todos los atributos apropiados para este tipo. Asimismo, los atributos SDO_ORDINATES y SDO_ELEM_INFO son VARRAYs, y son instanciados utilizando los tipos correspondientes, SDO_ELEM_INFO_ARRAY y SDO_ORDINATE_ARRAY, respectivamente.

El SDO_POINT puede almacenar solamente tres números, por tanto este atributo no puede almacenar puntos en cuatro dimensiones. Ejemplos de estos puntos incluyen localizaciones que almacenan temperaturas y altitudes. Para puntos de cuatro dimensiones, se necesita utilizar los atributos SDO_ELEM_INFO y SDO_ORDINATES del SDO_GEOMETRY. Sea (Xa, Ya, Za, La) las ordenadas de un punto de cuatro dimensiones. El SDO_GEOMETRY se rellena como se muestra en el Ejemplo 18. Hay que tener en cuenta que el único cambio en la columna geom, comparado con el Ejemplo 17, es que el atributo SDO_ORDINATES tiene cuatro números (que corresponden a las cuatro dimensiones), frente a dos del Ejemplo 17.

```
SQL> INSERT INTO geometry_examples VALUES
(
  '4-D POINT',
  '4-dimensional Point at (Xa=>2, Ya=>2, Za=>2, La=>2) con srid establecido a  NULL',
  SDO_GEOMETRY
  (
    4001,       -- SDO_GTYPE: D00T. Se establece a 4001 ya que es un punto de 4 dimensiones.
    NULL,      -- SDO_SRID
    NULL,      -- SDO_POINT:_TYPE is null
    SDO_ELEM_INFO_ARRAY (1,1,1),       -- Indica que es un punto
    SDO_ORDINATE_ARRAY(2, 2, 2, 2)     -- Aquí almacena las 4 coordenadas
   )
);
```

Ejemplo 18. Ejemplo de un Punto de Cuatro Dimensiones

Cadena de Línea: Conectado por Líneas Rectas

Se consideran los tres puntos A, B y C que se muestran en la Imagen 11. Para representar la línea que conecta estos tres puntos, se considera primero líneas rectas.

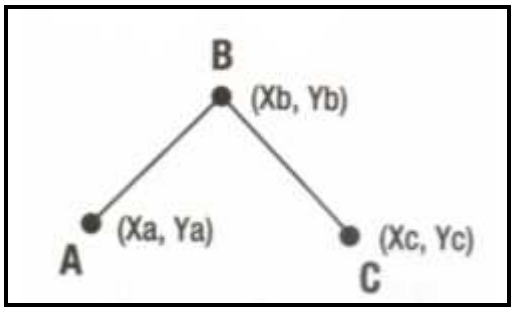


Imagen 11. Ejemplo de una cadena de línea conectada por líneas rectas

El objeto SDO_GEOEMTRY puede ser almacenado como se muestra en el Ejemplo 19.

```
SQL> INSERT INTO geometry_example VALUES
(
  'LINE STRING',
  '2-D line string connecting A(Xa=>1, Ya=>1), B(Xb=>2, Yb=>2), C(Xx=>2, Yx=>1)',
  SDO_GEOMETRY
  (
    2002,       -- SDO_GTYPE: D00T. Se establece a  2002 ya que es una línea de dos dimensiones
    32774,      -- SDO_SRID
    NULL,       -- SDO_POINT_TYPE es null
    SDO_ELEM_INFO_ARRAY       -- SDO_ELEM_INFO (ver los valores en la Tabla 3)
    (
      1,        -- Offset is 1
      2,        -- Element-type es 2 para una LINEA
      1         -- Interpretation es 1 si la línea esta conectada por líneas rectas.
    ),
    SDO_ORDINATE_ARRAY        -- SDO_ORDINATES
    (
      1,1,      -- valores Xa, Ya
      2,2,      -- valores Xb, Yb
      2,1       -- valores Xc, Yc
    )
  )
);
```

Ejemplo 19. Ejemplo de Línea de dos dimensiones

Dado que la geometría es una cadena de línea conectada por líneas restas, el atributo SDO_ELEM_INFO se establece a (1,2,1), como se describe en la Tabla 2. El atributo SDO_ORDINATES se rellena con las ordenadas de cada uno de los tres vértices A, B y C en el orden en que aparecen en la cadena de línea.

Se observa que todos los segmentos de línea son contiguos. Si se quiere almacenar líneas que no comparten vértices, se puede modelar usando geometrías de múltiples cadenas de línea.

Si hay N puntos con coordenadas (X1, Y1) … (XN, YN), y todos ellos necesitan ser conectados por líneas rectas en el orden (X1, Y1), (X2, Y2), …, (XN, YN), lo que hay que hacer es almacenar estos vértices en el atributo SDO_ORDINATES (en el orden en que deben ser conectados). El constructor de la geometría es como se muestra en el Ejemplo 20:

```
SDO_GEOMETRY
(
  2002, 32774, NULL,
  SDO_ELEM_INFO_ARRAY (1,2,1),
  SDO_ORDINATE_ARRAY (X1, Y1, X2, Y2, …, XN, YN)
)
```

Ejemplo 20. Ejemplo de almacenamiento de vértices

Cadena de línea: Conectada por Arcos

El ejemplo de la Imagen 12 almacena una cadena de línea compuesta por tres puntos. Sin embargo, esos mismos tres puntos pueden representar una forma muy diferente: un arco circular que para por los tres puntos.

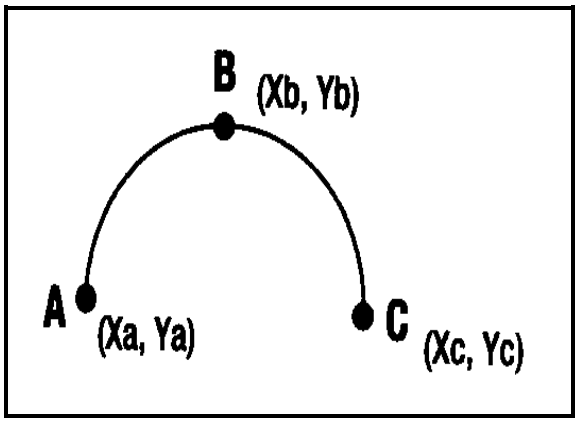


Imagen 12. Ejemplo de una cadena de línea conectada por arcos

Para hacer esto, simplemente hay que cambiar la interpretación en el SDO_ELEM_INFO a 2 (arco). El SDO_ELEM_INFO cambia de (1,2,1) a (1,2,2), como se muestra en el Ejemplo 21.

```
SQL> INSERT INTO geometry_examples VALUES
(
  'ARCSTRING',
  '2-D arc connecting A(Xa=>1, Ya=>1), B(Xb=>2, Yb=>2), C(Xx=>2, Yx=>1)',
  SDO_GEOMETRY
  (
    2002,       -- SDO_GTYPE: D00T. Se establece a 2002 como una línea de dos dimensiones
    32774,      -- SDO_SRID
    NULL,       -- SDO_POINT_TYPE es null
    SDO_ELEM_INFO_ARRAY       -- SDO_ELEM_INFO (ver los valores en la Tabla 3)
    (
      1,        -- Offset es 1
      2,        -- Element-type es 2 para una LINEA
      2         -- Interpretation es 2 si la línea es conectada por arcos.
    ),
    SDO_ORDINATE_ARRAY        -- SDO_ORDINATES
    (
      1,1,      -- valores Xa, Ya
      2,2,      -- valores Xb, Yb
      2,1       -- valores Xc, Yc
    )
  )
);
```

Ejemplo 21. Cadena de línea Conectada por Arcos en dos dimensiones

Si se quiere comparar esta representación con la del Ejemplo 18, solo hay que ver que la única diferencia es la interpretación (el tercer argumento en el SDO_ELEM_INFO_ARRAY), que ahora es 2. El resultado es una línea circular en vez de una línea recta.

Si la cadena de línea tiene más de tres puntos, el número de éstos debe ser impar. Un arco se construye con tres puntos, partiendo éste desde cada vértice impar (excepto el último vértice). Así que, si hay cinco puntos A, B, C, D y E, habrá dos arcos: el arco ABC que parte del vértice A y el arco CDE que parte del vértice C, como se muestra en la Imagen 13.

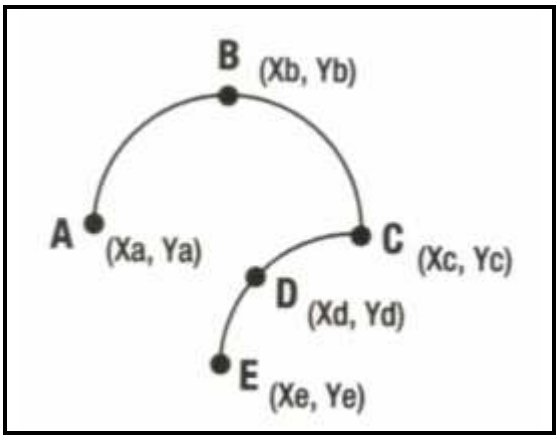


Imagen 13. Ejemplo de cadena de línea con múltiples arcos

El constructor para esta geometría es como se muestra en el Ejemplo 22:

```
SDO_GEOMETRY
(
  2002, 32774, NULL,
  SDO_ELEM_INFO_ARRAY (1,2,2),
  SDO_ORDINATE_ARRAY (Xa, Ya, Xb, Yb, Xc, Yc, Xd, Yd, Xe, Ye)
)
```

Ejemplo 22. Ejemplo de almacenamiento de vértices de un arco

Si se quiere modelar arcos que no sean contiguos, hay que considerar geometrías multilíneas/curvas.

Si la cadena de línea termina en el vértice de partida, provoca un bucle o un anillo. Pero no se puede considerar un polígono. Para ser considerado un polígono, el element-type en el atributo SDO_ELEM_INFO debe ser 1003 (o 2003).

Polígono: Anillo (Frontera) Conectado por Líneas Rectas

El límite (anillo) de un polígono puede ser conectado por líneas, arcos, o especificado como un rectángulo o un circulo.

La Imagen 14 muestra un ejemplo de un polígono donde el límite está conectado por líneas.

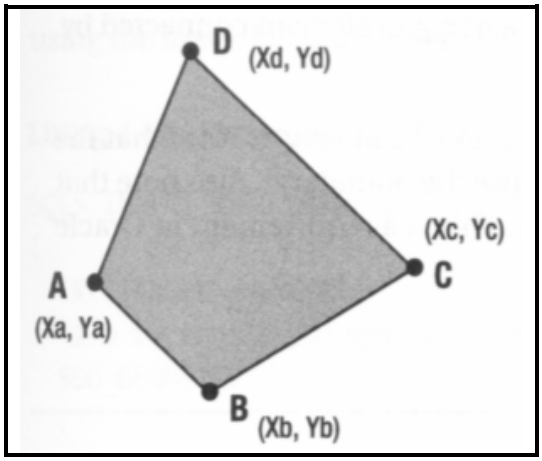


Imagen 14. Ejemplo de polígono conectado por líneas

El Ejemplo 23 muestra como insertar un polígono en la tabla geometry_examples.

```
SQL> INSERT INTO geometry_examples VALUES
(
'POLYGON',
'2-D polygon connecting A(Xa, Ya), B(Xb, Yb), C(Xc, Yc), D(Xd, Yd)',
SDO_GEOMETRY
(
2003,       -- SDO_GTYPE: D00T. Se establece a 2003 como un polígono de dos dimensiones
32774,      -- SDO_SRID
NULL,       -- SDO_POINT_TYPE es null
SDO_ELEM_INFO_ARRAY        -- SDO_ELEM_INFO (ver los valores en la Tabla 3)
(
1,        -- Offset is 1
1003,  -- Element-type es 1003 para un elemento exterior del POLIGONO
1         -- Interpretation es 1 si la frontera esta conectada por líneas
),
SDO_ORDINATE_ARRAY        -- SDO_ORDINATES
(
1,1,        -- valores Xa, Ya
2, -1,      -- valores Xb, Yb
3,1,        -- valores Xc, Yc
2,2,        -- valores Xd, Yd
1,1         -- valores Xa, Ya: repetir el primer vértice para cerrar el anillo
)
)
);
```

Ejemplo 23. Ejemplo de un polígono Conectado por Líneas

En comparación con los ejemplos anteriores, las principales diferencias son las siguientes:

- El SDO_GTYPE es 2003 (polígono de dos dimensiones).
- El element-type en el atributo SDO_ELEM_INFO es 1003, indica que es un polígono, y la interpretation es 1 para indicar que el polígono esta conectado por líneas rectas (ver la Tabla 3 como referencia).
- Las ordenadas del polígono son almacenadas en el atributo SDO_ORDINATES. Hay que tener en cuenta que el primer vértice (Xa, Ya) es repetido como ultimo vértice (para cerrar el limite).

También hay que tener en cuenta que los vértices se especifican en el orden contrario. Esto es un requisito de Oracle Spatial.

Polígono: Anillo (Frontera) Conectado por Arcos

El Ejemplo 23 se puede modificar fácilmente si se quiere modelar un polígono en el que cada tres vértices consecutivos estén conectados por arcos. Para ello, simplemente con cambiar la interpretation en el atributo SDO_ELEM_INFO a 2 ya estaría. Para que esto sea válido, se necesita un número impar de vértices. Sin embargo, para polígonos circulares, rara vez se utiliza en una representación de datos espaciales.

Rectángulo

Un rectángulo puede ser modelado como un polígono con cuatro vértices conectados por líneas rectas como en el ejemplo anterior. Sin embargo, una representación simplificada es posible especificando 1 instancia en vez de 3 para el interpretation en el SDO_ELEM_INFO. La Imagen 15 muestra un ejemplo de un rectángulo.

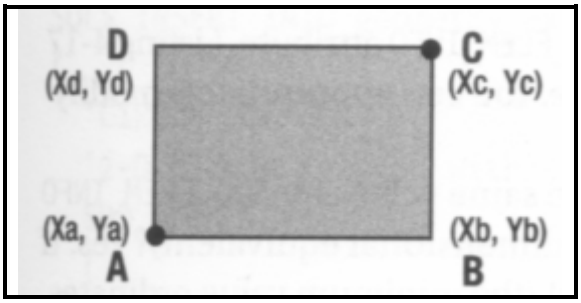


Imagen 15. Ejemplo de un rectángulo

La diferencia de este rectángulo con respecto a la Imagen 14, es que éste necesita solo dos vértices que se especificaran en lugar de los cuatro (es decir, es una representación mucho mas compacta): Oracle Spatial utiliza el vértice inferior izquierdo y el vértice superior derecho.

- El vértice de la esquina inferior izquierda tiene los valores mínimos de las ordenadas x e y. En la Imagen 15, A es el vértice de la esquina inferior izquierda.
- El vértice de la esquina superior derecha tiene los valores máximos de las ordenadas en x e y. En la Imagen 15, C es el vértice de la esquina superior derecha.

El Ejemplo 24 muestra como insertar un triangulo en la tabla geometry_examples utilizando los vértices inferior izquierda y superior derecha.

```
SQL> INSERT INTO geometry_examples VALUES
(
  'RECTANGLE POLYGON',
  '2-D rectangle polygon with corner points A(Xa, Ya), C(Xc, Yc)',
  SDO_GEOMETRY
  (
    2003,       -- SDO_GTYPE: D00T. Se establece a  2003 como un polígono de dos dimensiones
    32774,      -- SDO_SRID
    null,       -- SDO_POINT_TYPE es null
    SDO_ELEM_INFO_ARRAY       -- SDO_ELEM_INFO  (ver los valores de la Tabla 3)
    (
      1,       -- Offset es 1
      1003,  -- Element-type es 1003 para un POLIGONO (exterior)
      3        -- Interpretation es 3 si el polígono es un RECTANGULO
    ),
    SDO_ORDINATE_ARRAY       -- SDO_ORDINATES
    (
      1,1,       -- valores Xa, Ya
      2,2        -- valores Xc, Yc
    )
  )
);
```

Ejemplo 24. Ejemplo de un polígono Rectangular

Hay que tener en cuenta que la interpretation se establece a 3 en el atributo SDO_ELEM_INFO. El Ejemplo 22 especifica solo dos vértices en el atributo SDO_ORDINATES.

Un rectángulo en tres dimensiones es un paralelepípedo. Los mismos valores del SDO_ELEM_INFO pueden ser utilizados para representar un rectángulo en tres dimensiones. Si se tiene un paralelepípedo con el vértice inferior izquierdo en (Xa, Ya, Za) (el valor mínimo de las coordenadas X, Y, Z) y el vértice superior derecho en (Xc, Yc, Zc) (el valor máximo de las coordenadas X, Y, Z), entonces la geometría se parece a la siguiente. El SDO_GTYPE cambia de 2003 a 3003. Se puede construir el SDO_GEOMETRY para el rectángulo equivalente en cuatro dimensiones de manera análoga como se muestra en el Ejemplo 25:

```
SDO_GEOMETRY
(
  3003,       -- SDO_GTYPE se establece a 3003 para indicar un polígono de tres dimensiones.
  32774,
  NULL,
  SDO_ELEM_INFO_ARRAY(1, 1003, 3),
  SDO_ORDINATE_ARRAY(Xa, Ya, Za, Xc, Yc, Zc)
)
```

Ejemplo 25. Ejemplo de almacenamiento de un rectángulo en cuatro dimensiones

Círculo

La Imagen 16 muestra un ejemplo de un círculo. Los círculos son diferentes de los polígonos lineales y de los arcos solamente en la interpretation en el atributo SDO_ELEM_INFO y en el número de ordenadas del array SDO_ORDINATES. La interpretation se establece a 4, y el array almacena tres puntos distintos de la circunferencia del círculo.

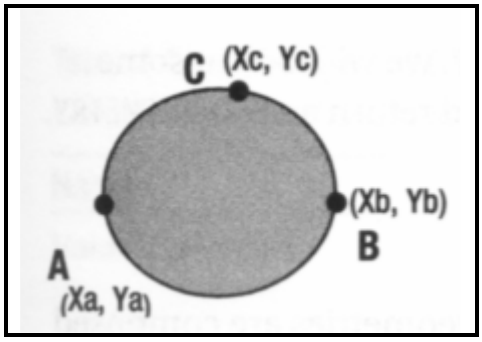


Imagen 16. Ejemplo de un polígono circular

El Ejemplo 26 muestra como insertar el polígono circular en la tabla geometry_example.

```
SQL> INSERT INTO geometry_examples VALUES
(
  'CIRCLE POLYGON',
  '2-D circle polygon with 3 coundary points A(Xa, Ya), B(Xb, Yb), C(Xc, Yc)',
  SDO_GEOMETRY
  (
    2003,       -- SDO_GTYPE: D00T. Se establece a 2003 como un polígono de dos dimensiones
    32774,      -- SDO_SRID
    NULL,       -- SDO_POINT_TYPE es null
    SDO_ELEM_INFO_ARRAY       -- SDO_ELEM_INFO (ver los valores de la Tabla 3)
    (
      1,        -- Offset es 1
      1003,  -- Element-type es 1003 para un POLIGONO (exterior)
      4         -- Interpretation es 4 si el polígono es un CIRCULO
    ),
    SDO_ORDINATE_ARRAY        -- SDO_ORDINATES
    (
      1,1,        -- valores Xa, Ya
      3,1,        -- valores Xb, Yb
      2,2         -- valores Xc, Yc
    )
  )
);
```

Ejemplo 26. Ejemplo de polígono Circular

2.10.4 Ejemplos de Geometría Compleja

En comparación con las geometrías simples, las geometrías complejas tienen más de un triplete descriptor para un elemento. Una geometría compleja puede ser:

- Una cadena de línea compuesta o un polígono compuesto: En la geometría de este tipo, el límite es conectado por líneas rectas y arcos circulares. Por ejemplo: las calles que tienen segmentos de líneas rectas y arcos (para denotar que conecta las carreteras) puede almacenarse como una composición de la geometría de cadena de línea. Los objetos G y F en la Imagen 7 son ejemplos de una cadena de línea compuesta y de un polígono compuesto, respectivamente.
- Un polígono vacío: Esta geometría tiene un anillo exterior y uno o más anillos interiores. Los elementos del anillo interior y exterior del polígono son especificados como elementos de un polígono simple. El objeto D en la Imagen 7 es un ejemplo de una geometría de polígono vacío. Los lagos y otras masas de agua que tienen islas pueden ser almacenadas como polígonos vacíos. Hay que tener en cuenta que la zona de los anillos interiores no se considera parte de estas geometrías.
- Una colección: esta geometría es una colección de múltiples elementos como puntos, líneas y/o polígonos. El objeto E en la Imagen 8 es un ejemplo de una colección. Los límites del estado de Texas y California tienen una o más islas y se pueden almacenar como una geometría de colección.

Directrices para la Construcción de Geometrías Complejas

Para construir geometrías complejas, estas son las directrices generales:

- Los tripletes del SDO_ELEM_INFO de los elementos simples que constituyen la geometría compleja se concatenan en el orden apropiado.
- Los valores del SDO_ORDINATES también con concatenados. (No se duplican los vértices compartidos de los elementos contiguos).
- Como resultado de la concatenación del SDO_ORDINATES, los offsets en el atributo SDO_ELEM_INFO para cada elemento simple se ajustan para reflejar el inicio correcto del elemento en el array del SDO_ORDINATES.
- Para los elementos compuestos (cadena de línea o polígono), los tripletes adicionales se añaden en el SDO_ELEM_INFO para especificar la combinación de los siguientes elementos simples. La Tabla 4 muestra los posibles valores que puede tomar el elemet-type para estos tripletes adicionales.
- El SDO_GTYPE se fija en función de la geometría resultante.

Nombre	Element-Type	Interpretation
Polígono Vacío	1003 = Polígono Exterior 2003 = Polígono Interior	1 = Limite del polígono conectado por líneas rectas. 2 = Limite del polígono conectado por arcos. 3 = Polígono rectángulo. Los vértices de la esquina inferior izquierda y de la esquina superior derecha del rectángulo se especifican en el array del SDO_ORDINATES. 4 = Polígono circular. Tres vértices del límite del círculo son especificados en el array SDO_ORDINATES.
Cadena de Línea Compuesta	4	N = Especifica el numero de subelementos que constituyen la cadena de línea compuesta. Los N tripletes para estos N subelementos siguen el mismo triplete
Polígono compuesto	1005 = Polígono exterior 2005 = Polígono Interior	N = Especifica el numero de subelementos de líneas rectar y arcos que constituyen el limite del polígono. Los N tripletes para esto N suplementos siguen este triplete.

Tabla 4. Valores para <Element-type, Interpretation> en el Elemento del Triplete Descriptor para las Geometrías de Polígonos Compuestos/Vacíos.

SDO_ELEM_INFO para Elementos Compuestos

Si el elemento compuesto tiene N subelementos, entonces habrá N + 1 tripletes descriptores: un triplete encabezado que especifica que se trata de un elemento compuesto, seguido de N tripletes, uno por cada subelemento.

Los N subelementos tienen que ser elementos simples, y sus tripletes descriptores se construirán según lo especificado anteriormente para los elementos simples. El triplete encabezado tiene la siguiente forma:

- El offset especifica la posición inicial del elemento compuesto en el array SDO_ORDINATES.
- El element-type especifica una de las siguientes particularidades:
    - Una cadena de línea compuesta (element-type = 4).
    - Un polígono compuesto (element-type = 1005 ó 2005). El element-type será 1005 si el elemento compuesto es utilizado en un anillo de un polígono exterior, ó 2005 si es utilizado en un anillo interior.
- La interpretation especifica el número de subelementos que forman este elemento compuesto.

Por ejemplo, para la cadena de línea compuesta (objeto F en la Imagen 7), el element-type para el triplete encabezado será 4 y la interpretation será 2 ya que tiene dos subelementos. Los dos tripletes siguientes del array SDO_ELEM_INFO tendrán la descripción para esos subelementos. Ambos elementos son líneas y tienen el element_type igual a 2 pero un subelemento tendrá la interpretation igual a 1, indicando la conectividad de línea recta, y el otro elemento tendrá la interpretation igual a 2, indicando la conectividad en arco. El SDO_ELEM_INFO tendrá los tripletes en el orden siguiente:

- (1,4,2) para el triplete encabezado especificando la cadena de línea compuesta.
- (1,2,1) para el triplete del primer subelemento conectado por líneas rectas.
- (5,2,2) para el triplete del siguiente subelemento conectado por un arco.

SDO_ELEM_INFO para el Elemento del polígono Vacío

Si el polígono vacío tiene N subelementos vacíos (anillos interiores) y un subelemento como anillo exterior, entonces habrá al menos N + 1 tripletes descriptores. El primer triplete especificará el triplete descriptor para el anillo exterior. Éste irá seguido de los tripletes descriptores para cada uno de los N subelementos vacíos. Si todos los subelementos son elementos simples, entonces habrá exactamente N + 1 tripletes descriptores. De lo contrario, el tamaño se reflejará en los descriptores para los subelementos compuestos.

Por ejemplo, el polígono vacío (objeto D en la Imagen 7) tiene dos tripletes descriptores. El primer triplete representa el anillo exterior del polígono y tiene el element-type igual a 1003. El segundo triplete representa el rectángulo vacío y tiene el element-type igual a 2003.

A continuación, se muestran algunos ejemplos de tales formas complejas y la forma de representarlos utilizando el tipo de datos SDO_GEOMETRY en Oracle.

Ejemplo de Cadena de Línea Compuesta

La mayoría de los segmentos de una carretera están conectados por líneas rectas. Sin embargo, hay algunos sectores donde la carretera toma una forma de giro circular. La Imagen 17 muestra un ejemplo.

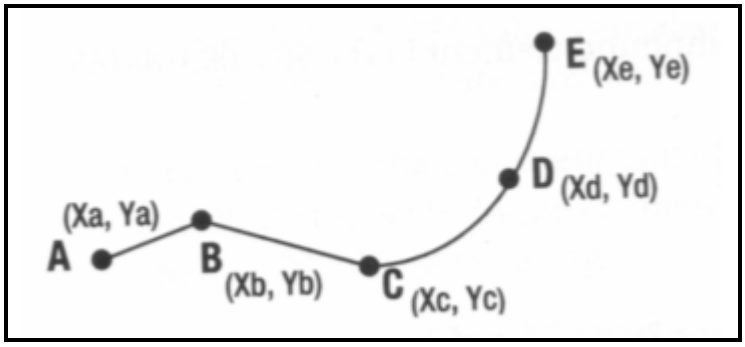


Imagen 17. Ejemplo de una cadena de línea compuesta conectada por líneas y arcos

El segmento de línea ABC esta conectado por líneas rectas, y CDE esta conectado por arcos. Para representar esta línea compuesta. Hay que construir un elemento compuesto para especificar un triplete encabezado seguido de tripletes de elementos simples para el atributo SDO_ELEM_INFO como se describe en la tabla 4.

- Triplete encabezado: El número de subelementos es 2 y el desplazamiento inicial es 1, por lo que el triplete encabezado es (1, 4, 2). El valor 4 especifica que es in triplete encabezado para una cadena de línea compuesta, y el valor 2 especifica que el número de elementos simples es dos.
- El triplete para la línea ABC: Ya que éste es el primer elemento simple, el offset en el atributo SDO_ORDINATES seguirá siento 1. El element-type se establece a 2 (cadena de línea) y la interpretation se establece a 1 para indicar la conectividad en línea recta. Este triplete es (1,2,1). Las seis ordenadas para este elemento son almacenadas primero en el array SDO_ORDINATES.
- El triplete para el arco CDE: Además, este elemento parte del vértice C con el elemento anterior ABC, así las ordenadas para el vértice C no es necesario que se repitan. Dado que este elemento (CDE) comienza en el vértice C, se almacena en la quinta posición, la posición inicial se establece a 5. El element-type se establece a 2 (cadena de línea) y la interpretation se establece también a 2 para indicar la conectividad en arco. El triplete por lo tanto es (5,2,2).

Dado que la geometría solo tiene líneas en dos dimensiones, el SDO_GTYPE se establece a 2002. Esta representación se describe utilizando los elementos del SDO_GEOMETRY en la Imagen 18.

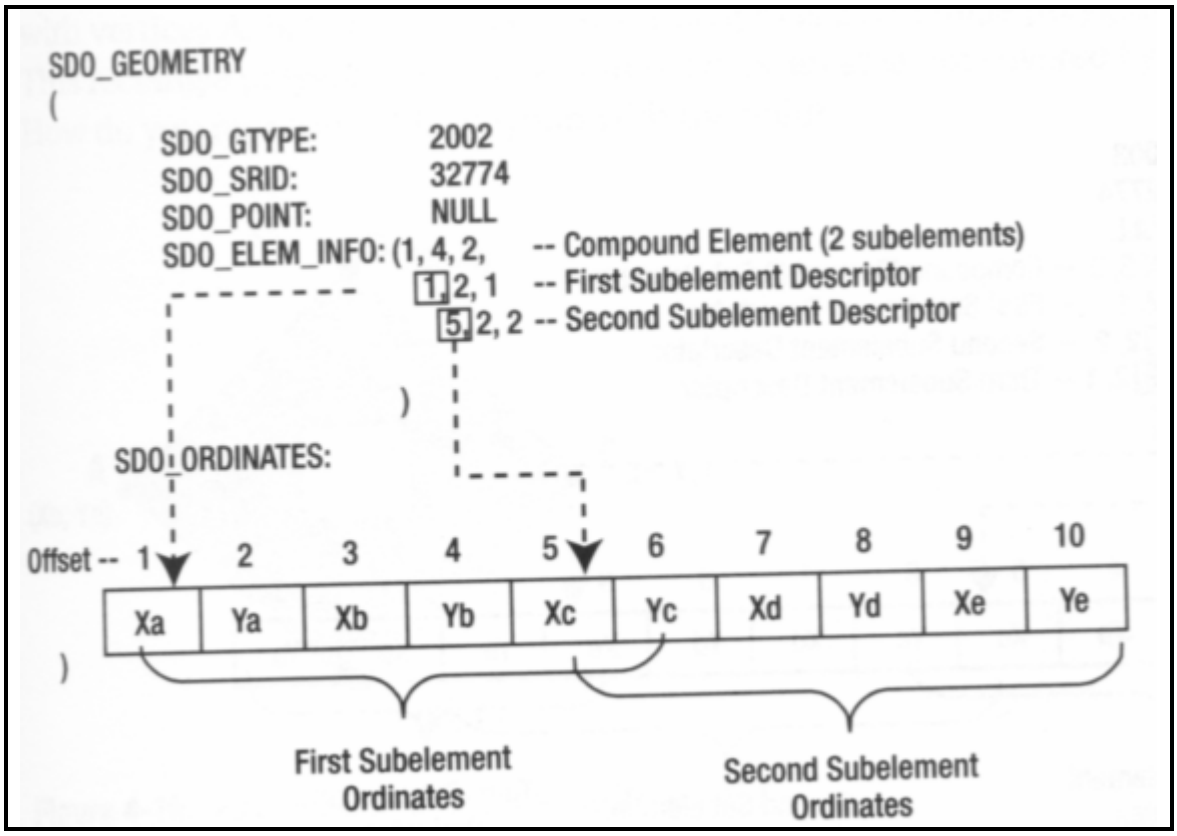


Imagen 18. Almacenando una cadena de línea compuesta como un SDO_GEOMETRY

Ejemplo de Polígono Compuesto

Si se quiere conectar el vértice E con el vértice A en la Imagen 17, hay que convertir la línea en una línea cerrada como se muestra en la Imagen 19.

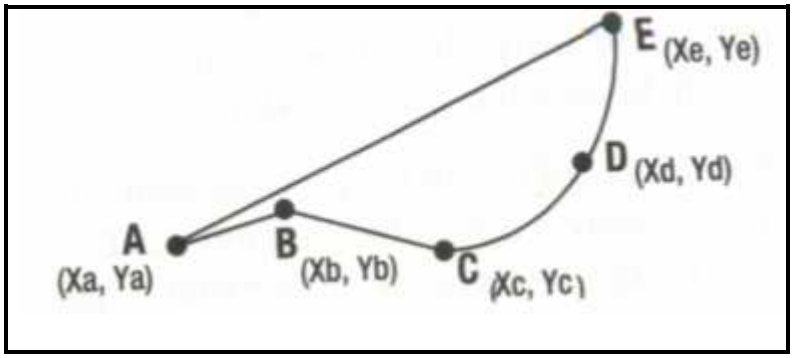


Imagen 19. Ejemplo de una cadena de línea compuesta "cerrada" conectada por líneas y arcos

Se puede utilizar esta cadena de línea compuesta cerrada de forma ademada como limites de un polígono modificando el atributo SDO_GTYPE (y el element-types). La Imagen 20 muestra los elementos para el SDO_GEOMETRY.

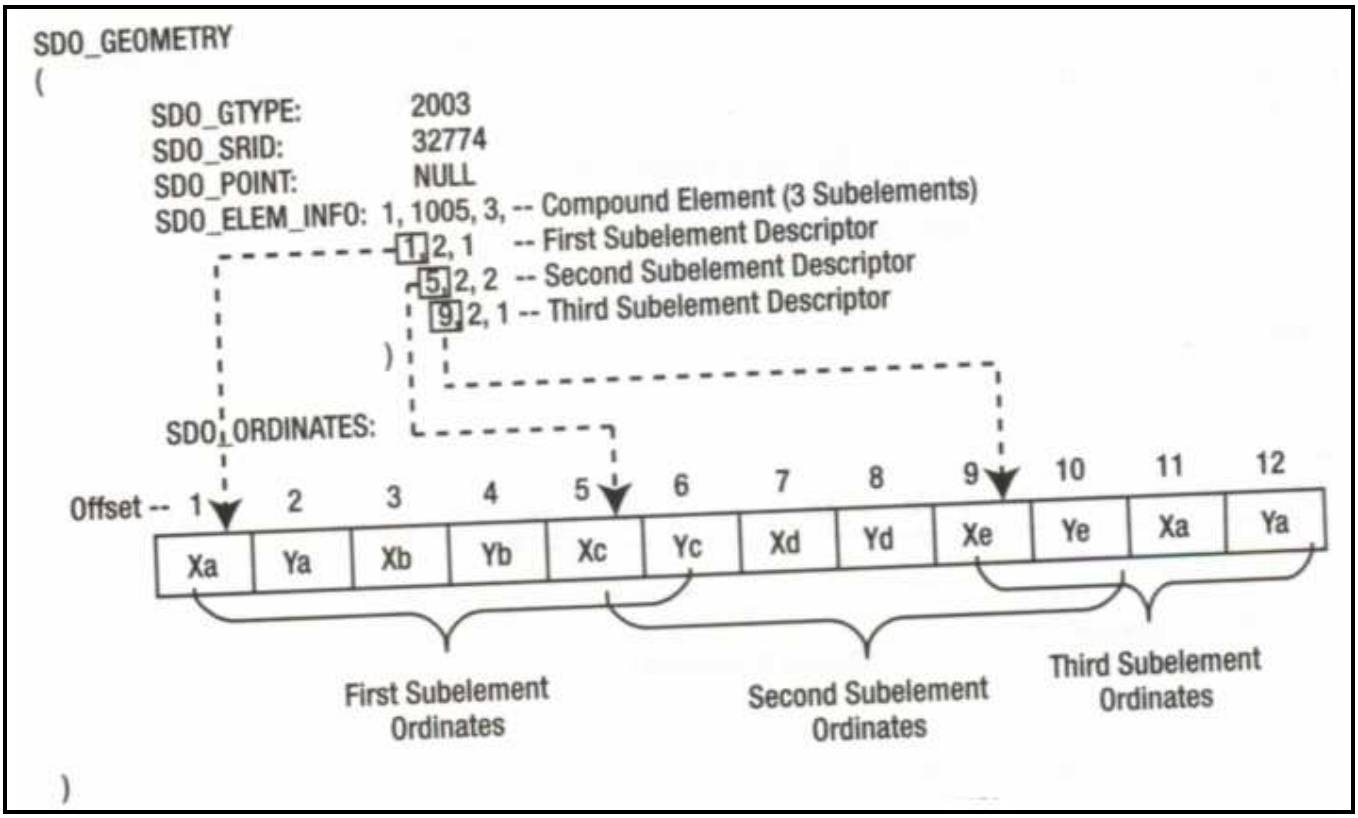


Imagen 20. Almacenando un polígono compuesto como un SDO_GEOMETRY

Estos son los cambios:

- El SDO_GTYPE se establece a 2003.
- Para el triplete encabezado, el element-type se establece a 1005 (polígono compuesto) en lugar de 4, y el número de suplementos cambia de dos a tres.
- Un nuevo subelemento representa la línea recta que conecta E con A. Este subelemento tiene el SDO_ELEM_INFO establecido a (9,2,1) donde 9 representa la posición inicial de las ordenadas de E, 2 indica que es una línea, y 1 indica la conectividad por línea recta.
- Las ordenadas para el vértice A son repetidas al final en el array SDO_ORDINATES.

Hay que recordar que, excepto para el triplete encabezado, todos los tripletes de los otros suplementos tiene el elment-type a 2 (línea), ya que estos elementos solo representan líneas. El triplete encabezado significa que el compuesto tiene un element-type a 1005.

Polígono con un vacío

Los polígonos con un vacío ayudan a representar los océanos que tienen islas. La Imagen 21 muestra un diamante con forma de polígono con vértices A, B, C y D. Dentro de este polígono hay un polígono rectángulo con esquinas E y F. Este polígono rectángulo sirve como vacío, es decir, un área no cubierta por el polígono exterior ABCD.

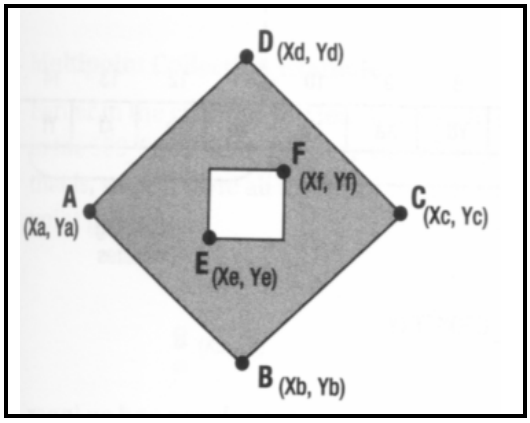


Imagen 21. Ejemplo de un polígono con un vacío.

Para representar este polígono con el vacío, hay que examinar los constructores de los dos polígonos de la Imagen 21 por separado. El polígono ABCD (sin vacío) es un polígono simple cuya frontera está conectada por líneas rectas. El constructor es el que se muestra en el Ejemplo 27:

```
SDO_GEOMETRY
(
  2003, 32774, NULL,
  SDO_ELEM_INFO(1, 1003, 1),
  SDO_ORDINATE_ARRAY(Xa, Ya, Xb, Yb, Xc, Yc, Xd, Yd, Xa, Ya)
)
```

Ejemplo 27. Ejemplo de almacenar un polígono con un vacío

Suponiendo que el polígono rectangular EF no esta dentro de ABCD, el constructor es el que se muestra en el Ejemplo 28:

```
SDO_GEOMETRY
(
  2003, 32774, NULL,
  SDO_ELEM_INFO(1, 1003, 3),
  SDO_ORDINATE_ARRAY(Xe, Ye, Xf, Yf)
)
```

Ejemplo 28. Ejemplo de almacenar el rectángulo de la Imagen 21

Usando estos dos constructores, se pueden combinar los dos polígonos para representar un polígono con un vacío como el que se muestra en la Imagen 21. En la Imagen 22, el descriptor del elemento externo describe el polígono exterior, y el descriptor del elemento interno describe el polígono interior.

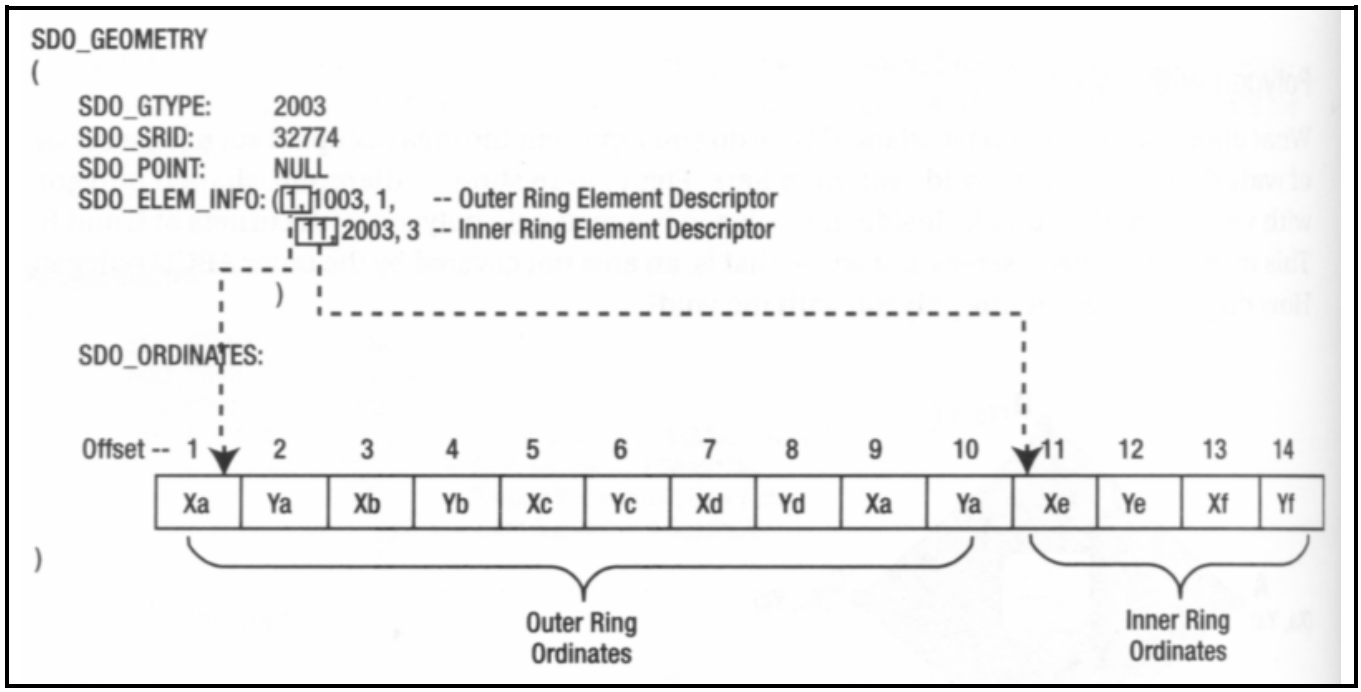


Imagen 22. Almacenando un polígono con un vacío como un SDO_GEOMETRY

En este ejemplo, el polígono combinado tiene dos elementos: un polígono exterior y un polígono interior. Es necesario especificar estos elementos de la siguiente manera:

- Los tripletes para el polígono exterior son especificados en primer lugar, seguidos por el polígono interior (es decir, el vacío). El anillo del polígono exterior debe tener el element-type establecido a 1003 y el anillo del polígono interior debe tener el element-type establecido a 2003. (Si hay más de un polígono interior, estos son especificados en cualquier orden después de la especificación del polígono exterior).
- Asimismo, las ordenadas del polígono exterior son especificadas primero, seguidas de las ordenadas del polígono interior.

- La posición inicial de las ordenadas del polígono interior se ajustan de 1 a 11 (ya que son precedidos por las ordenadas del polígono exterior).

Se puede tener un polígono dentro del vacío (es decir, dentro del anillo interior), pero se trata como una geometría multipolígono (el valor del atributo SDO_GTYPE es 7). La razón de esto, es que el área representada por el polígono resultando no es contigua.

Colecciones

Las colecciones pueden ser homogéneas, como un multipunto, multipolígono, una colección multipolígono. Ó pueden ser heterogéneas, conteniendo una combinación de geometrías de puntos, líneas, y/o polígonos. En la Tabla 2 se vió que cada multipunto, multilínea, multipolígono, y colecciones heterogéneas tienen un valor diferente para el SDO_GTYPE. Ahora se detallará como representar estas geometrías usando el tipo de dato SDO_GEOMETRY.

Ejemplo de Colección Multipunto

Se va a modelar múltiples puntos como una única colección geométrica –es decir, se almacenarán los tres puntos A, B y C de la Imagen 23 como subelementos de una única geometría multipunto.

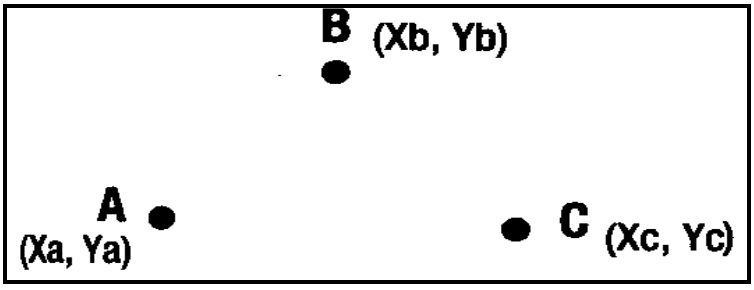


Imagen 23. Ejemplo de una colección multipunto

Para almacenar esta geometría, primero se construyen los objetos SDO_GEOMETRY para los puntos individuales y se combinan utilizando las directrices descritas anteriormente.

1. Se establece el SDO_GTYPE a 5 (multipunto).
2. Se combinan los atributos SDO_ORDINATES del objeto SDO_GEOMETRY de los tres puntos.
3. Se combinan los atributos correspondientes del SDO_ELEM_INFO de los tres puntos. El desplazamiento resultante en el SDO_ELEM_INFO es ajustado para reflejar el desplazamiento en el atributo SDO_ORDINATES para cada punto.

El resultado del SDO_GEOMETRY es el que se muestra en el Ejemplo 29:

```
SDO_GEOMETRY
(
  2005, 32774, NULL,
  SDO_ELEM_INFO_ARRAY     -- SDO_ELEM_INFO: varios elementos, cada uno con 1 punto
  (
    1,1,1,                -- triplete para el primer “punto” del elemento
    3,1,1,                -- triplete para el segundo “punto” del elemento
    5,1,1                 -- triplete para el tercer “punto” del elemento
  ),
  SDO_ORDINATE_ARRAY
  (
    Xa, Ya,               -- coordenadas del primer punto
    Xb, Yb,               -- coordenadas del segundo punto
    Xc, Yc                -- coordenadas del tercer punto
  )
)
```

Ejemplo 29. Almacenamiento de la geometría de la Imagen 23

En este ejemplo, los tres puntos son representados como tres elementos. Oracle, sin embargo, tiene una representación mucho mas simple: se pueden representar tres puntos como un simple elemento (y almacenar todas las coordenadas en el atributo SDO_ORDINATES). El elemento tendrá un triplete descriptor con la forma (1, 1, N) donde N representa el numero de puntos (si N = 1, entonces el elemento tiene un solo punto). El constructor correspondiente es el que se muestra en el Ejemplo 30, y los cambios están en negrita.

```
SDO_GEOMETRY
(
  2005, 32774, NULL,
  SDO_ELEM_INFO_ARRAY     -- SDO_ELEM_INFO
  (
    1, 1, 3               -- Elemento con 3 puntos
  ),
  SDO_ORDINATE_ARRAY
  (
    Xa, Ya,               -- coordenadas del primer punto
    Xb, Yb,               -- coordenadas del segundo punto
    Xc, Yc                -- coordenadas del tercer punto
  )
)
```

Ejemplo 30. Ejemplo de almacenar tres puntos como un simple elemento

Es recomendable el uso de un solo elemento de N puntos en lugar de un array de puntos. Esta representación es más eficiente en cuanto al almacenamiento y ayuda en el rendimiento.

Multilínea

Una geometría multilínea consiste en múltiples cadenas de línea. La Imagen 24 muestra un ejemplo. Los tripletes en el SDO_ELEM_INFO son utilizados para indicar y delimitar cada segmento de línea.

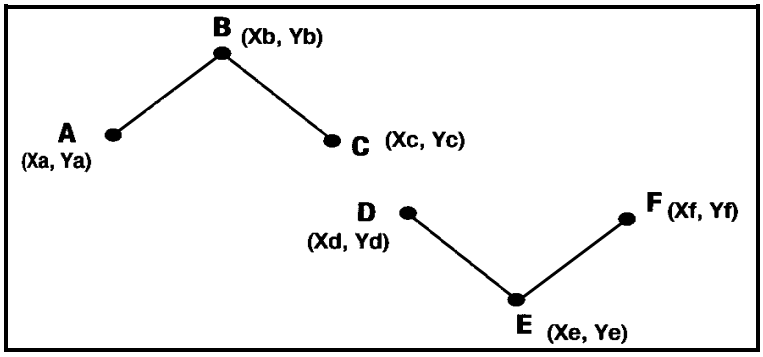


Imagen 24. Ejemplo de multilínea

Se utiliza un triplete en el array SDO_ELEM_INFO para representar cada uno de los elementos en la geometría. La Imagen 25 muestra el resultado del constructor SDO_GEOMETRY para una multilínea.

- La primera línea (ABC) comienza en la posición 1 y termina en la posición 6 (es decir, tiene dos ordenadas para cada uno de los tres puntos).
- La segunda línea (DEF) comienza en la posición 7 y termina en la posición 12 (es decir, tiene dos ordenadas para cada uno de los tres puntos).

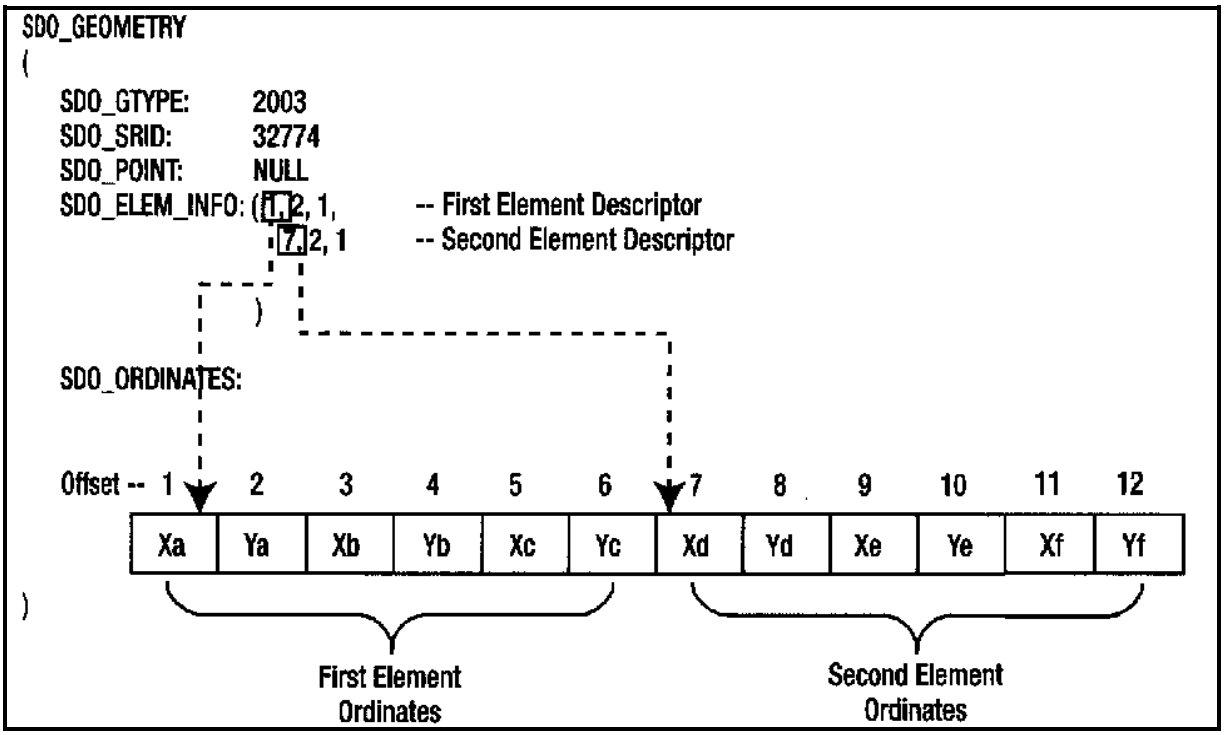


Imagen 25. Almacenando una multilínea en el SDO_GEOMETRY

Hay que tener en cuenta que los dos elementos geométricos pueden tener interpretaciones diferentes; una podría ser una línea recta, y la otra un arco circular. Hay que tener en cuenta también que si la geometría es tridimensional, las posiciones (excepto la primera) serían diferentes.

Multipolígono y Colecciones heterogéneas

En los casos de geometrías multipunto y colecciones multilínea, los tripletes en la estructura SDO_ELEM_INFO son utilizados para describir cada elemento de la colección. El SDO_GTYPE se establece con un valor apropiado para la colección. Las ordenadas de cada elemento de la colección so almacenadas en el array SDO_ORDINATES, y las posiciones de comienzo son registradas en el correspondiente triplete del SDO_ELEM_INFO para cada elemento de la colección. Este mecanismo se describe a continuación.

Creando colecciones: La Forma más Fácil

La función SDO_UTIL.APPEND toma dos geometrías superpuestas y devuelve una geometría anexada. Por ejemplo, si se invoca APPEND utilizando dos polígonos, se obtiene una geometría multipolígono como resultado. Hay que tener en cuenta que si la geometría de entrada se superpone, esta función puede devolver una geometría inválida, en el Ejemplo 31 se muestra como utilizar dicha función:

```
SQL> SELECT SDO_UTIL.APPEND
(
  SDO_GEOMETRY
  (
    2003, 32774, null,
    SDO_ELEM_INFO_ARRAY(1,1003,3),
    SDO_ORDINATE_ARRAY(1,1,  2,2)
  ),
  SDO_GEOMETRY
  (
    2003, 32774, NULL,
    SDO_ELEM_INFO_ARRAY(1, 1003, 3),
    SDO_ORDINATE_ARRAY(2,3,  4,5)
  )
)
FROM dual;

SDO_UTIL.APPEN(SDO_GEOMETRY(2003,32774,NULL,…
- - - - - - - - - - - - --- -- - - - - - - - - - - - - - - - - - - - - - - -
SDO_GEOMETRY
(
  2007,    -- SDO_GTYPE = Multi-Poligono
  32774, NULL,
  SDO_ELEM_INFO_ARRAY(1, 1003, 3, 4, 1003, 3),
  SDO_ORDINATE_ARRAY(1, 1, 2, 2, 2, 3, 4, 5)
)
```

Ejemplo 31. Geometría anexada a partir de dos geometrías superpuestas

Si se pasa una línea y un polígono, se obtiene una colección heterogénea (SDO_GTYPE = 2007), como en el Ejemplo 32:

```
SQL> SELECT SDO_UTIL.APPEND
(
  SDO_GEOMETRY
  (
    2003, 32774, null,
    SDO_ELEM_INFO_ARRAY(1,1003,3),
    SDO_ORDINATE_ARRAY(1,1,  2,2)
  ),
  SDO_GEOMETRY
  (
    2002, 32774, NULL,
    SDO_ELEM_INFO_ARRAY(1, 2, 2),
    SDO_ORDINATE_ARRAY(2,3,  3,3,  4,2)
  )
)
FROM dual;

SDO_UTIL.APPEN(SDO_GEOMETRY(2003,32774,NULL,…
- - - - - - - - - - - - --- -- - - - - - - - - - - - - - - - - - - - - - - --
SDO_GEOMETRY
(
  2004,    -- SDO_GTYPE = Colección (Heterogénea)
  32774, NULL,
  SDO_ELEM_INFO_ARRAY(1, 1003, 3, 5, 2, 2),
  SDO_ORDINATE_ARRAY(1, 1, 2, 2, 2, 3, 4, 2)
)
```

Ejemplo 32. Colección heterogénea a partir de una línea y un polígono

2.11 Conceptos generales de la red de modelado

A continuación se definen unos conceptos genéricos para la creación de redes.

Red: tipo de gráfico matemático que capta las relaciones entre los objetos usando la conectividad. Se compone de nodos y enlaces.

- Un nodo representa un objeto de interés en la red. Para una red de carreteras, los nodos son las intersecciones.
- Un enlace representa una relación entre dos nodos. Dichos componentes tienen una serie de restricciones:

  - Cada enlace une dos y sólo dos nodos.
  - Múltiples enlaces pueden contener el mismo nodo.
  - Dos nodos pueden estar conectados por múltiples enlaces.
  - Los enlaces pueden ser dirigidos o no dirigidos.
  - Un enlace no dirigido puede ser recorrido en cualquier dirección, mientras que un enlace dirigido permite moverse en una sola dirección.

- Los nodos pueden definir la dirección de un enlace: un enlace dirigido se considera un "flujo" desde su nodo inicio hasta su nodo fin.
- El enlace de un enlace dirigido es el enlace que une "los flujos" entre los mismos nodos, en la dirección opuesta.

En una red de carreteras, los enlaces representan las carreteras y segmentos de calles entre intersecciones. Un enlace dirigido representa una calle de sentido único. En la Imagen 26, los enlaces L1 y L2 representan calles de sentido único. Los enlaces L3 y L4 representan calles de doble sentido.

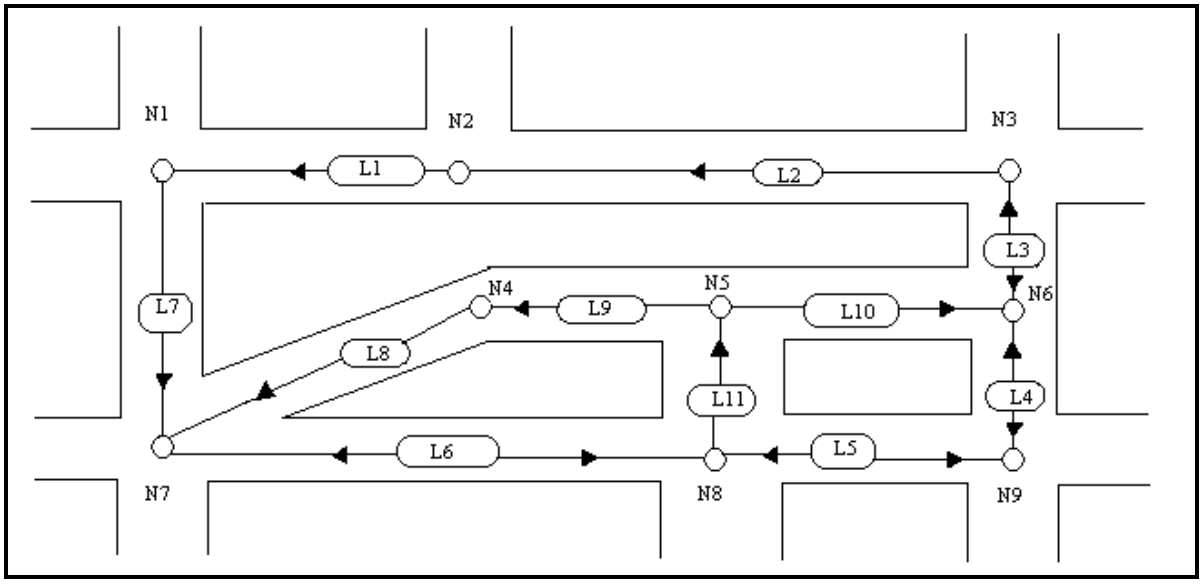


Imagen 26. Componentes de la red: nodos y enlaces

Los elementos de la red (nodos y enlaces) pueden tener información geométrica asociada a ellos. Una red contiene la información de la conectividad y la información geométrica.

En Oracle Espacial, en una red espacial, los nodos y enlaces son objetos SDO_GEOMETRY que representan puntos y líneas, respectivamente.

- Un camino representa una ruta a través de la red. Características:
  - Está formado por una secuencia de nodos y enlaces entre dos nodos.
  - Puede haber múltiples caminos entre dos nodos.
  - Una ruta puede ser simple o compleja.
  - En un camino simple, los enlaces de una lista ordenada se pueden recorrer del nodo inicio al nodo fin, con cada uno de los enlaces visitados una vez.
  - Un camino complejo representa una subred entre un nodo de inicio y destino.

  La Imagen 26 muestra un camino del nodo N7 al nodo N6, pasando a través de los enlaces L6, L10 y L11.

- El coste es un atributo de tipo numérico, al cual que se pueden asociar con enlaces o nodos. El coste se utiliza para el cálculo de rutas entre dos nodos: el coste de un camino es la suma de los costes de todos los nodos y enlaces en ese camino. La ruta de coste mínimo es el camino que tiene el menor coste total desde un nodo inicio a un nodo fin. Por ejemplo, la distancia o el tiempo más corto. Los enlaces y nodos pueden tener múltiples costes, dependiendo del camino que pase por ellos (distintos caminos pueden pasar por un mismo enlace y nodo), pero sólo uno se utiliza a la vez.

En una red de carreteras, el coste de una conexión suele ser la longitud de la carretera o el segmento de carretera representado por ese enlace. Esto es bueno para el cálculo de la ruta más corta entre dos puntos. La mayoría de las redes de carreteras también incluyen el tiempo de conducción a lo largo de la carretera. Esto se utiliza para calcular la ruta más rápida entre dos lugares.

Por último, las limitaciones de la red son las restricciones definidas en las búsquedas de red. En una red de carreteras, por ejemplo, las rutas puede ser necesarias para incluir sólo las carreteras que sean accesibles para los camiones o para evitar vías de peaje. Otras restricciones pueden estar basadas en el tiempo, tales como cierres de paso de montaña durante el invierno, la operación de transbordadores, o la prohibición durante las horas de mayor tráfico.

2.12 Estructura de datos: Red de tablas

Para poder tener en cuenta toda la información comentada en el apartado anterior, el sistema gestor de base de datos que se va a utilizar en este proyecto (Oracle 11g) define una serie de tablas.

Principalmente, una red se define con dos tablas: una tabla de nodos y una tabla de enlaces, no obstante, este SGBD también incorpora una tabla de caminos y otra de enlaces-caminos. Éstas últimas son opcionales y contienen los resultados de los análisis realizados en API Java, como el camino más corto entre dos nodos.

La Imagen 27 muestra las relaciones entre las tablas que describen una red:

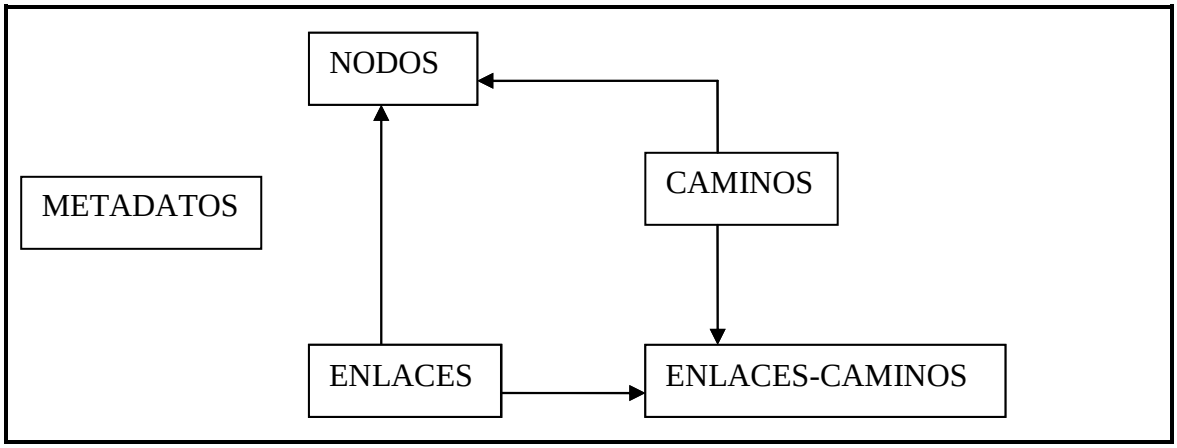


Imagen 27. Tablas de una Red

Las tablas pueden ser nombradas como queramos, pero deben seguir una estructura bien definida. O, para más precisión, deben contener un número mínimo de columnas, algunas con nombres predefinidos.

Puede haber flexibilidad en la estructuración de las tablas - algunas columnas se pueden llamar de cualquier manera (las columnas de geometría y coste, en particular), y su orden no es importante. También se puede incluir más columnas para tener información adicional. Y, por último, las tablas pueden ser vistas de tablas existentes.

El nombre de las tablas que constituyen una red y su estructura se define en una tabla de metadatos llamada USER_SDO_NETWORK_METADATA, que actualiza de la misma manera la base espacial de metadatos (USER_SDO_METADATA). En particular, ahí es donde se especifica el nombre de la columna de las tablas nodos y enlaces que define el valor del coste, o el nombre de la columna de geometría. Esto también permite definir múltiples redes en el mismo conjunto de tablas, utilizando diferentes columnas de coste - por ejemplo, una red basada en las distancias y otra basada en los tiempos de viaje.

Existen varias técnicas para la definición de las estructuras de datos para una red. Por una parte, una simple llamada a in procedimiento hace todo: crea todas las tablas con los nombres predeterminados y rellena los metadatos. Por otra parte, crear todas las tablas manualmente (o crear vistas sobre las tablas existentes) y rellenar manualmente los metadatos. Las funciones se utilizan para verificar que las estructuras de los datos son válidos y coherentes.

A continuación, se describe la estructura de cada una de las tablas de la red. Para cada tabla, se indican las columnas requeridas y las opcionales. Las columnas tienen el nombre en minúscula y puede tener cualquier nombre. Su nombre real es definido en los metadatos.

2.12.1 Tabla Nodo

La tabla nodo describe todos los nodos de la red. Cada nodo tiene un único identificador numérico (columna ID_nodo). Esta es la única columna obligatoria, todas las demás son opcionales.

Columna	Tipo de dato	Descripción
ID_NODO	Number	Identificación única para ese nodo en la red. Es también la clave primaria de la red
GEOMETRIA	SDO_GEOMETRY	Punto que contiene las coordenadas del nodo. Solo para redes espaciales. Las redes lógicas no contienen geometrías.
COSTE	Number	Representa el coste de atravesar ese nodo. Puede haber múltiples costes asociados a un nodo. Cuando ésta columna no es definida, se asume que el coste para todos los nodos es 0.
NIVEL_JERARQUIA	Number	Para redes jerárquicas solamente. Es el nivel del nodo.
PADRE_NODO	Number	Para redes jerárquicas solamente. Es el identificador del nodo padre de este nodo.
ACTIVO	Char(1)	Define si el nodo esta activo (visible en la red o no) – ‘Y’ o ‘N’. Cuando esta columna no es definida, se asume que todos los nodos están activos.
NOMBRE_NODO	Varchar2(32)	Nombre del nodo. Contiene un nombre descriptivo.
TIPO	Varchar2(24)	Tipo de nodo. Contiene un código o nombre descriptivo.

Tabla 5. La tabla del Nodo

2.12.2 Tabla Enlace

La tabla enlace describe todos los enlaces de la red. Cada enlace tiene un único identificador numérico (columna ID_enlace) y contiene los identificadores de los dos nodos que conecta. Todas las otras columnas son opcionales.

Columna	Tipo de dato	Descripción
ID_ENLACE	Number	Identificación única para ese enlace en la red. Es también la clave primaria de la red
ID_NODO_INICIO	Number	Identificación única del nodo inicio del enlace
ID_NODO_FIN	Number	Identificación única del nodo fin del enlace
GEOMETRIA	SDO_GEOMETRY	Línea que describe la forma del enlace. Solo para redes espaciales. Las redes lógicas no contienen geometrías
COSTE	Number	Representa el coste de atravesar ese enlace. Puede haber múltiples costes asociados a un enlace. Cuando ésta columna no es definida, se asume que el coste para todos los enlaces es 1.
PADRE_ENLACE	Number	Para redes jerárquicas solamente. Es el identificador del nodo padre de este enlace.
ACTIVO	Char(1)	Define si el enlace esta activo (visible en la red o no) – 'Y' o 'N'. Cuando esta columna no es definida, se asume que todos los nodos están activos.
NIVEL_ENLACE	NUMBER	Prioridad en el enlace.
NOMBRE_ENLACE	Varchar2(32)	Nombre del enlace. Contiene un nombre descriptivo.
TIPO	Varchar2(24)	Tipo de enlace. Contiene un código o nombre descriptivo.

Tabla 6. La tabla del Enlace

2.12.3 Tabla Camino

La tabla camino contiene los nodos de inicio y fin de un camino, y su coste total. La columna coste es obligatoria. La lista de los enlaces que describen un camino esta en la tabla enlace-camino. Las tablas camino y enlace-camino son opcionales – es necesario crearlas ya que se van a conservar los caminos calculados por el análisis de redes API Java para poder realizar búsquedas mas rápido en caso de que dos usuarios quieran ir al mismo destino.

Columna	Tipo de dato	Descripción
ID_CAMNO	Number	Identificación única para ese camino en la red. Es también la clave primaria de la red.
ID_NODO_INICIO	Number	Identificación única del nodo inicio del camino
ID_NODO_FIN	Number	Identificación única del nodo fin del camino
COSTE	Number	Valor numérico que representa el coste total del camino.
SIMPLE	Char(1)	Contiene Y si el camino es camino simple o N si este es complejo. Si la columna no existe, entonces todos los caminos son considerados simples. Todos los caminos producidos por Java API son simples.
GEOMETRIA	SDO_GEOMETRY	Línea que describe el tipo de camino, formado por enlaces. Solo para redes espaciales.
NOMBRE_ENLACE	Varchar2(32)	Nombre del camino. Contiene un nombre descriptivo.
TIPO	Varchar2(24)	Tipo del camino. Contiene un código o nombre descriptivo.

Tabla 7. La tabla del Camino

2.12.4 Tabla Enlace-Camino

La tabla Enlace-Camino, almacena todos los enlaces que definen un camino. El ID_CAMINO e ID_ENLACE forman la clave primaria de la tabla. Todas las columnas son obligatorias.

Columna	Tipo de dato	Descripción
ID_CAMINO	Number	Identificación del camino
ID_ENLACE	Number	Identificación del enlace
SEQ_NO	Number	Secuencia del enlace en el camino

Tabla 8. La tabla del Enlace-Camino

2.12.5 Red Metadatos

La vista USER_SDO_GEOM_METADATA, describe los elementos que componen una red: los nombres de las tablas y los nombres de las columnas opcionales así como los costes y las geometrías.

Nombre	Tipo de dato	Descripción
RED	Varchar2(24)	Nombre único de red. Esta limitado a 24 caracteres
ID_RED	Number	Numero único de red (opcional)
CATEGORIA_RED	Varchar2(12)	La categoría de red es ESPACIAL si los nodos y enlaces de la red son asociados con geometrías espaciales y LOGICA si los nodos y enlaces de la red no son asociados con geometrías espaciales.
TIPO_GEOMETRIA	Varchar2(24)	El tipo de geometría espacial si la categoría de la red es ESPACIAL. Esto es normalmente establecido a SDO_GEOMETRY pero también puede ser fijado a LRS_BEOMETRY o TOPO_GEOMETRY.
TIPO_RED	Varchar2(24)	Cadena definida por el usuario para describir el tipo de red
NUM_NIVELES_JERARQUIA	Number	Numero de niveles en la jerarquía de la red. Contiene un 1 si solo tiene un nivel.
NUM_PARTICIONES	Number	Numero de particiones en la red. Contiene un 1.
DIRECCION_ENLACE	Varchar2(12)	Especifica si los enlaces de la red son dirigidos o no.
NOMBRE_TABLA_NODO	Varchar2(32)	Nombre de la tabla Nodo
COLUM_GEOM_NODO	Varchar2(32)	Nombre de la columna de geometría en la tabla nodo (si la categoría de la red es ESPACIAL).
COLUM_COTE_NODO	Varchar2(32)	Nombre de la columna de coste en la tabla nodo. Si no se especifica, entonces en el análisis de la red no se usa ningún coste del nodo. (Ej. Todos los nodos

		tienen coste 0)
NOMBRE_TABLA_ENLACE	Varchar2(32)	Nombre de la tabla Enlace
COLUM_GEOM_ENLACE	Varchar2(32)	Nombre de la columna de geometría en la tabla enlace (si la categoría de la red es ESPACIAL).
COLUM_COTES_ENLACE	Varchar2(1024)	Nombre de la columna de coste en la tabla enlace. Si no se especifica, entonces en el análisis de la red no se usa ningún coste del enlace. (Ej. Todos los enlaces tienen coste 1).
NOMBRE_TABLA_CAMINO	Varchar2(32)	Nombre de la tabla camino. Es opcional. Si no se especifica, entonces la red no usa ninguna tabla camino.
NOMBRE_TABLA_ENLACE_CAMINO	Varchar2(32)	Nombre de la tabla enlace_camino. Es opcional. Solamente se especifica si la red usa una tabla camino.
COLUM_GEOM_CAMINO	Varchar2(32)	Nombre de la columna de geometría en la tabla camino.
NOMBRE_TABLA_LRS	Varchar2(32)	Nombre de la tabla que contiene las geometrías LRS (solamente cuando el tipo de geometría es LRS_GEOMETRY).
COLUM_GEOM_LRS	Varchar2(32)	Nombre de la columna de geometría en la tabla LRS.
NOMBRE_TABLA_PARTICION	Varchar2(32)	Actualmente no se utilizan.

Tabla 9. La vista USER_SDO_GEOM_METADATA

La red más simple posible es la que contiene una tabla nodo solamente con la columna ID_NODO, una tabla enlace que contiene las columnas ID_ENLACE, ID_NODO_INICIO, ID_NODO_FIN. Esta es una red lógica, no direccionada y sin costes.

2.13 Definiendo Redes

Como se mencionó anteriormente, hay varias formas de definir las estructuras de datos para una red. Por un lado, una simple llamada a un procedimiento hará todo. Y por otro lado, se crean todas las tablas manualmente. Todas las operaciones son proporcionadas por los procedimientos y las funciones del paquete SDO_NET.

2.13.1 Definición de Red "Automáticamente"

Se puede usar el procedimiento CREATE_SDO_NETWORK o CREATE_LOGICAL_NETWORK para crear todas las estructuras de una red.

En el Ejemplo 33, se crea una red espacial llamada US_ROADS. Los enlaces son dirigidos, y los nodos no tienen coste.

Los nombres no se han facilitado para las diversas tablas; recibirán por defecto nombres generados. En este ejemplo, se usa la notación *named* para los parámetros. Esta notación es más detallada, pero hace que su código sea más fácil de leer y mantener. Es especialmente útil para llamar a procedimientos o funciones que tienen una larga lista de parámetros.

```
SQL> BEGIN
SDO_NET.CREATE_SDO_NETWORK (
NETWORK => 'US_ROADS',
NO_OF_HIERARCHY_LEVELS => 1,
IS_DIRECTED => TRUE,
NODE_WITH_COST => FALSE
);
END;
```

Ejemplo 33: Creando una Red Espacial usando por defecto nombres de tablas

La llamada puede reducirse como sigue (usando el parámetro de la notación posicional):

```
SQL> EXEC SDO_NET.CREATE_SDO_NETWORK ('US_ROADS' ,1,TRUE,FALSE)
```

El procedimiento crea cuatro tablas para la red llamadas US_ROADS_NODE$, US_ROADS_LINK$, US_ROADS_PATH$, y US_ROADS_PLINK$, y añade información a la red metadatos. La columna de geometría en las tablas node, link, y path link es llamada GEOMETRY. La tabla link tiene una columna de coste llamada COST. Las tablas path y path link son creadas, incluso si no queremos. El Ejemplo 34 muestra este procedimiento:

```
SQL> describe US_ROADS_NODE$
Name                          Null?                 Type
--------------------------    ------------------    ----------------------
NODE_ID                                              NOT NULL
NODE_NAME                                             VARCHAR2(32)
NODE_TYPE                                             VARCHAR2(24)
ACTIVE                                               VARCHAR2(1)
PARTITION_ID                                          NUMBER
GEOMETRY
```

```
SQL> describe US_ROADS_LINK$
Name                               Null?            Type
--------------------------    ------------------    ----------------------
LINK _ID                                            NOT NULL
LINK _NAME                                           VARCHAR2(32 CHAR)
START_NODE_ID                                        NOT NULL
END_NODE_ID                                         NOT NULL
LINK _TYPE                                           VARCHAR2(24 CHAR)
ACTIVE                                               VARCHAR2(1 CHAR)
LINK_LEVEL                                           NUMBER
GEOMETRY                                              MDSYS.SDO_GEOMETRY
COST                                                  NUMBER
```

```
SQL> describe US_ROADS_PATH$
Name                          Null?                 Type
--------------------------    ------------------    ----------------------
PATH _ID                                            NOT NULL
PATH _NAME                                           VARCHAR2(32 CHAR)
PATH_TYPE                                            VARCHAR2(24 CHAR)
START_NODE_ID                                        NOT NULL
END_NODE_ID                                          NOT NULL
COST                                                 NUMBER
SIMPLE                                                VARCHAR2(1 CHAR)
GEOMETRY                                             MDSYS.SDO_GEOMETRY
```

```
SQL> describe US_ROADS_PLINK$
Name                           Null?                 Type
--------------------------    ------------------    ----------------------
PATH _ID                                             NOT NULL
LINK_ID                                              NOT NULL
SEQ_NO
```

Ejemplo 34: Estructura de las tablas por defecto de la red

Las tablas tienen definidas las claves primarias (usando nombres por defecto). Sin embargo, las claves ajenas no son definidas.

En el Ejemplo 35 se crea la misma red US_ROADS con los nombres de las tablas y las columnas explícitos. También muestra la lista completa de los parámetros que se puede pasar al procedimiento.

```
SQL> BEGIN
SDO_NET.CREATE_SDO_NETWORK (
NETWORK => 'US_ROADS',
NO_OF_HIERARCHY_LEVELS => 1,
IS_DIRECTED => TRUE,
NODE_TABLE_NAME => 'US_INTERSECTIONS',
NODE_GEOM_COLUMN => 'LOCATION',
NODE_COST_COLUMN => NULL,
LINK_TABLE_NAME => 'US_STREETS',
LINK_GEOM_COLUMN => 'STREET_GEOM',
LINK_COST_COLUMN => 'STREET_LENGTH',
PATH_TABLE_NAME => 'US_PATHS',
PATH_GEOM_COLUMN => 'PATH_GEOM',
PATH_LINK_TABLE_NAME => 'US_PATH_LINKS'
);
END;
```

Ejemplo 35: Creación de red con nombres de tablas y columnas explícitos

Cuando no se le da nombre a la columna geometría, su nombre es GEOMETRY. Cuando no se le da nombre a la columna del coste, dicha columna no se crea.

2.13.2 Definición de Red "Manual"

El método "automático" de creación no es flexible. Da muy poco control sobre la estructuración de las tablas, y no da el control en todo su almacenamiento físico (espacio de tablas, la gestión del espacio, particionamiento, etc.) Pero es fácil de usar. En particular, se rellenan automáticamente los metadatos de la red y se asegura que las estructuras de las tablas son coherentes.

La alternativa es crear la red manualmente. Esto da una flexibilidad total en las estructuras de las tablas, pero deberá actualizarse la red metadato manualmente y garantizar que las tablas son coherentes con las estructuras de los metadatos.

Para nuestra red, la crearemos de forma manual como veremos mas adelante en apartado de base de datos

2.13.3 Definiendo múltiples Redes en una misma tabla

En el Ejemplo 36, también se puede calcular la ruta más rápida entre los nodos. Para ello, todo lo que se necesita es definir una segunda red en la misma tabla, esta vez utilizando la columna del coste travel_time para los enlaces. Se define una nueva red llamada US_ROADS_TIME.

```
SQL> INSERT INTO USER_SDO_NETWORK_METADATA (
NETWORK,
NETWORK_CATEGORY,
GEOMETRY_TYPE,
NO_OF_HIERARCHY_LEVELS,
NO_OF_PARTITIONS,
LINK_DIRECTION,
NODE_TABLE_NAME,
NODE_GEOM_COLUMN,
NODE_COST_COLUMN,
LINK_TABLE_NAME,
LINK_GEOM_COLUMN,
LINK_COST_COLUMN,
PATH_TABLE_NAME,
PATH_GEOM_COLUMN,
PATH_LINK_TABLE_NAME
)
VALUES (
'US_ROADS_TIME',          -- red (calve primaria)
'SPATIAL',                -- categoría de red
'SDO_GEOMETRY',           -- tipo geometría
1,
1,
'DIRECTED',
'US_INTERSECTIONS',       -- nombre tabla nodo
'LOCATION',               -- columna geometría nodo
NULL,
'US_STREETS',             -- nombre tabla enlace
'STREET_GEOM',            -- columna geometría enlace
'TRAVEL_TIME',            -- columna coste enlace
' US_PATHS',              -- nombre tabla camino
'PATH_GEOM',              -- columna geometría camino
'US_PATH_LINKS'           -- nombre tabla enlace-camino
);
SQL>COMMIT;
```

Ejemplo 36: La creación de metadatos para un tiempo basado en la red de carreteras

2.13.4 Definiendo una red a través de las estructuras existentes

En el Ejemplo 36, se han definido las estructuras de la red totalmente de cero. ¿Qué pasa si ya hay una red definida (como tablas de nodo y de enlace) y se utilizan en las aplicaciones existentes?

Una forma es crear una copia de la red existente en las tablas adecuadas para el modelo de datos de red de Oracle. Este enfoque tiene inconvenientes:

- Duplica los gastos de almacenamiento.
- Añade complejidad al mantenimiento de las dos copias de sincronización la red.

Un enfoque más sencillo consiste en definir una red directamente en las tablas existentes, sólo para la creación de los metadatos de la red para que apunten a las tablas existentes. Esto, sin embargo, no es posible directamente. Aunque la estructura de las tablas de red es flexible, aún existen algunas limitaciones - por ejemplo, la clave primaria en la tabla nodo debe ser llamada NODE_ID.

La solución es crear vistas sobre las tablas existentes, y cambiar el nombre de las columnas en las vistas para que coincidan con los nombres convencionales de las tablas de la red. Esto se muestra en el Ejemplo 37. Se considera la posibilidad de una aplicación existente que ya utiliza una red de distribución de agua, que se define como un conjunto de tuberías y válvulas. Esas tablas fueron creadas por la aplicación.

```
SQL> CREATE TABLE valves (
Valve_id                    NUMBER PRIMARY KEY,
Valve_type                    VARCHAR2(20),
Location                    SDO_GEOMETRY
-- … otras columnas …
);
SQL> CREATE TABLE pipes (
Pipe_id                     NUMBER PRIMARY KEY,
Diameter                          NUMBER,
Length                            NUMBER,
Start_valve        NUMBER NOT NULL REFERENCES VALVES,
End_valve          NUMBER NOT NULL REFERENCES VALVES,
Pipe_geom                   SDO_GEOMETRY
-- … other columns …
);
```

Ejemplo 37: Existencia de tablas de la red de distribución de agua

Primer paso: definir las vistas de las tablas existentes, como se muestra en el Ejemplo 38. Podemos observar que la columna valve_id en la tabla válvulas original es renombrado a node_id en la vista net_pipes. Un cambio de nombre similar tiene lugar en la vista net_pipes.

```
SQL> CREATE VIEW net_valves (node_id, valve_type, location) AS
SELECT valve_id,
Valve_type,
Location
 -- … otras columnas …
FROM valves;

SQL> CREATE VIEW net_pipes
(link_id, start_node_id, end_node_id, length, pipe_geom)
AS
SELECT pipe_id,
Start_valve,
End_valve,
Length,
Pipe_geom
-- … otras columnas …
FROM  pipes;
```

Ejemplo 38: Vistas de las tablas existentes

El segundo paso es opcional. Si queremos mantener los resultados de cualquier análisis o trazas de red, entonces tenemos que crear también las tablas camino y enlace camino (ver ejemplo). Hay poca flexibilidad en la creación de las tablas - todo lo que podemos hacer es elegir el nombre de las tablas y elegir un nombre para la columna SDO_GEOMETRY de la tabla camino. El Ejemplo 39 muestra este caso:

```
SQL> CREATE TABLE net_paths (
Path_id                         NUMBER,
Start_node_id                NUMBER NOT NULL,
End_node_id                  NUMBER NOT NULL,
Cost                                                    NUMBER,
Simple                                                  VARCHAR2(1),
Path_geom                     SDO_GEOMETRY,
CONSTRAINT net_paths_pk PRIMARY KEY (path_id)
);

SQL> CREATE TABLE net_paht_links (
Path_id                         NUMBER,
Link_id                         NUMBER,
Seq_no                          NUMBER,
CONSTRAINT net_path_links_pk PRIMARY KEY (path_id, link_id)
);
```

Ejemplo 39: Creación de las tablas camino y enlace-camino

El paso final es la creación de la red de metadatos para la red de distribución de agua. Se muestra en el Ejemplo 40. Se puede observar que esta red no es direccionada - en una tubería, el agua puede fluir en cualquier dirección.

```
SQL> INSERT INTO USER_SDO_NETWORK_METADATA (
NETWORK,
NETWORK_CATEGORY,
GEOMETRY_TYPE,
NO_OF_HIERARCHY_LEVELS,
NO_OF_PARTITIONS,
LINK_DIRECTION,
NODE_TABLE_NAME,
NODE_GEOM_COLUMN,
NODE_COST_COLUMN,
LINK_TABLE_NAME,
LINK_GEOM_COLUMN,
LINK_COST_COLUMN,
PATH_TABLE_NAME,
PATH_GEOM_COLUMN,
PATH_LINK_TABLE_NAME
)
 VALUES (
'WATER_NET',                -- red (clave primaria)
'SPATIAL',                  -- categoria de la red
'SDO_GEOMETRY'              -- tipo de geometria
1,
1,
'UNDIRECTED',               -- direccion de los enlaces
'NET_VALVES',               -- nombre de la tabla nodo
'LOCATION',                 -- columna geometrica del nodo
NULL,                       --columna coste del nodo
'NET_PIPES',                -- nombre de la tabla enlace
'PIPE_GEOM',                -- columna geometrica del enlace
'LENGTH',                   -- columna coste del enlace
'NET_PATHS'                 -- nombre de la tabla camino
'PATH_GEOM',                -- columna geometrica del camino
'NET_PATH_LINKS'            -- nombre de la tabla enlace-camino
;
SQL> COMMIT;
```

Ejemplo 40: Metadata para una red existente en estructura (la estructura ya existe)

2.13.5 Validación de estructuras de la red

Se pueden cometer errores en la definición de todas las estructuras de la red y metadatos al crearla manualmente. Es recomendable verificar la exactitud de su definición utilizando una de las funciones predefinidas.

La función principal es VALIDATE_NETWORK. Verifica la coherencia entre los metadatos y las tablas de la red, y verifica que las tablas estén correctamente definidas (es decir, que contienen columnas con el correspondiente tipo de datos). La función toma un solo argumento: el nombre de la red para validar.

El resultado de la función es una cadena que puede tomar los siguientes valores:

- NULL si la red no existe.
- TRUE si la red está correctamente definida.
- Una cadena con el diagnóstico si la red no está correctamente definida.

2.13.6 Caída de una red

Para excluir a una red, hay que utilizar el procedimiento DROP_NETWORK. Dejará caer todas las tablas relacionadas con una red y eliminar la definición de la red de los metadatos. El Ejemplo 41 muestra la caída de una red:

```
SQL> EXEC SDO_NET.DROP_NETWORK ('US_ROADS');
PL/SQL procedure successfully completed.
```

Ejemplo 41: Caída de una red

El procedimiento localiza las tablas que componen una red, como se indica en la red de metadatos. Si los metadatos son incorrectos - por ejemplo, si contiene el nombre equivocado de la tabla nodo - entonces el procedimiento intenta abandonar la tabla cuyo nombre se encuentra en los metadatos. Hay que tener en cuenta que también se elimina automáticamente cualquier forma de metadatos espaciales USER_SDO_GEOM_METADATA.

2.13.7 Índices espaciales en las tablas de red

Para redes espaciales, con tablas que contienen la columna SDO_GEOMETRY, puede ser necesario crear índices espaciales. Esto no es un requisito; las funciones de análisis de redes no necesitan índices espaciales.

Los índices espaciales necesitan metadatos espaciales en la tabla USER_SDO_GEOM_METADATA. Se pueden insertar manualmente los metadatos, o podemos utilizar el procedimiento INSERT_GEOM_METADATA para establecer los metadatos para todas las tablas que forman parte de la red y que contienen una columna SDO_GEOMETRY. En el Ejemplo 42 se puede ver. La función toma el nombre de la red como un parámetro de entrada, seguido de los límites y las definiciones SRID.

```
SQL> BEGIN
SDO_NET.INSERT_GEOM_METADATA (
'US_ROADS',
SDO_DIM_ARRY (
SDO_DIM_ELEMENT ('Long', -180, +180, 1),
SDO_DIM_ELEMENT ('Lat', -90, +90, 1)
),
8307
);
END;
/
SQL> COMMIT;
```

Ejemplo 42: Añadiendo metadatos espaciales a las tablas de Red

2.13.8 Obtener información sobre una red

Para saber los detalles sobre la estructura de una red (el nombre de la tabla de enlace, el nombre de la columna de gastos en la tabla nodo, etc.), sólo es necesario la vista USER_SDO_NETWORK_METADATA. Hay que tener en cuenta que la salida puede ser difícil de leer.

Podemos usar una de las muchas funciones del paquete SDO_NET que devuelve la información. Los nombres de las funciones son auto-descriptivos. Todos ellos tienen un argumento: el nombre de la red a examinar.

2.13.9 Verificar la conectividad de la Red

El paquete SDO_NET proporciona algunas funciones, se resumen en la Tabla 10, que nos ayudarán a localizar los errores dentro de la red de datos como nodos aislados o enlaces colgando. Todas las funciones toman el nombre de red como parámetro.

Función	Descripción
SDO_NET.GET_NO_OF_NODES()	Devuelve el número de nodos en la Red.
SDO_NET.GET_NO_OF_LINKS()	Devuelve el número de enlaces en la Red.
SDO_NET.GET_ISOLATED_NODES()	Devuelve los nodos que no están relacionados con ningún enlace.
SDO_NET.GET_INVALID_LINKS()	Devuelve los enlaces con nodos de inicio o fin inexistentes.
SDO_NET.GET_INVALID_PATHS()	Devuelve los caminos con nodos de inicio o fin inexistentes, o con enlaces inexistentes.

Tabla 10: Funciones de verificación de la Red

Algunas otras funciones, que figuran en Tabla 11, permiten averiguar los detalles sobre los nodos de la red. Todas ellas toman dos parámetros: el nombre de la red y el identificador del nodo a examinar.

Función	Descripción
SDO_NET.GET_NODE_DEGREE()	Devuelve el número de enlaces que empiezan y terminan en ese nodo.
SDO_NET.GET_NODE_IN_DEGREE()	Devuelve el número de enlaces que terminan en ese nodo.
SDO_NET.GET_NODE_OUT_DEGREE()	Devuelve el número de enlaces que empiezan en ese nodo.
SDO_NET.GET_IN_LINKS()	Devuelve una lista de IDs con todas las notas que terminan en ese nodo.
SDO_NET.GET_OUT_LINKS()	Devuelve una lista de IDs con todas las notas que empiezan en ese nodo.

Tabla 11: Detalles de las funciones de los nodos

3. Análisis y Diseño

A fin de facilitar la comprensión de este proyecto, se van a aplicar los conceptos introducidos para obtener la solución del problema propuesto. Para ello se va a comenzar con el análisis del problema, mostrando los objetivos del sistema, y el diseño de éste.

Por tanto, en este apartado se va a explicar la arquitectura de la aplicación, como paso previo, para después diseñar la base de datos que cubre las necesidades de almacenamiento y manejo de la información que forma parte del sistema.

3.1 Arquitectura

La arquitectura del sistema se va a diseñar en función de los objetivos prefijados anteriormente, y también con otros objetivos como la mantenibilidad, flexibilidad e interacción con otros sistemas de información. Esto último se va a tener en cuenta porque se quiere crear un sistema con posibilidad de uso futuro en otro tipo de aplicaciones (con las modificaciones pertinentes).

Por lo tanto, teniendo en cuenta que los objetivos que se persiguen alcanzar son:

- Realizar búsquedas de caminos en un mismo nivel, es decir, cuando el usuario está en el nivel 1 y quiere ir a un punto del nivel 1, o si está en el nivel 0 y quiere ir a un punto del nivel 0.
- Realizar búsquedas de caminos en distintos niveles, es decir, si el usuario esta en el nivel 0 y quiere ir a un punto del nivel 1 o viceversa.

La arquitectura que se obtiene es la que se muestra en la Imagen 28, en la cual se definen los distintos componentes que la forman, así como la comunicación entre ellos para alcanzar la funcionalidad global.

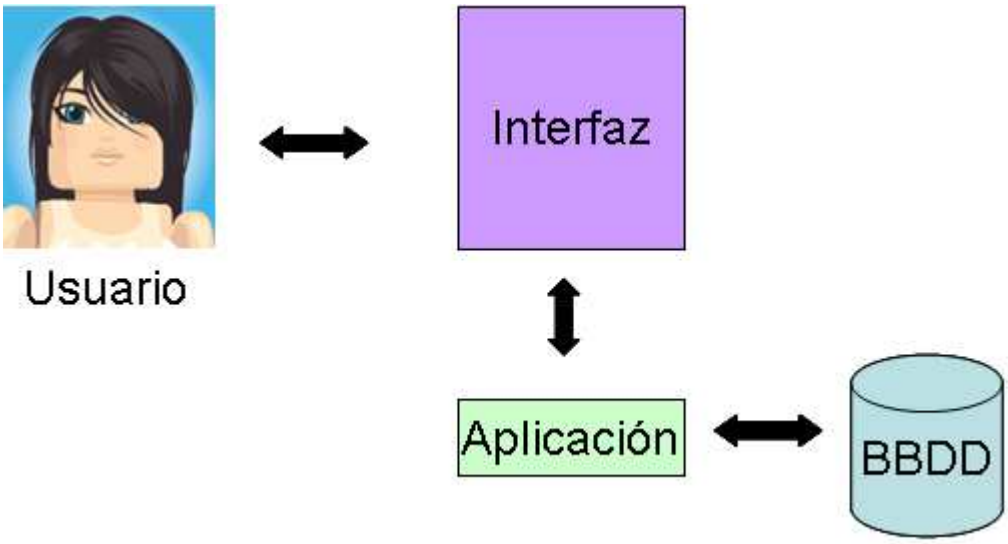


Imagen 28. Esquema Funcional de la aplicación

A continuación de describe cada uno de sus componentes:

- Interfaz: contiene todo lo necesario para permitir al usuario una interacción rápida y sencilla entre éste y la información que maneja la aplicación desarrollada. Es decir, va a encargarse de:

    - Obtener la información introducida por el usuario y pasársela a la aplicación.
    - Recoger la información aportada por la aplicación, y tras darle el formato adecuado, mostrársela al usuario.

- Aplicación: A través de la información recogida por el usuario, va a realizar cualquier búsqueda del camino desde el lugar en el que se encuentra el usuario al deseado por el mismo. Es decir, resuelve la solicitud del usuario mostrando la ruta más adecuada.

  El diseño de como va a ser dicha aplicación, es decir, que pasos se van a seguir para obtener el resultado final se muestra en la Imagen 29.

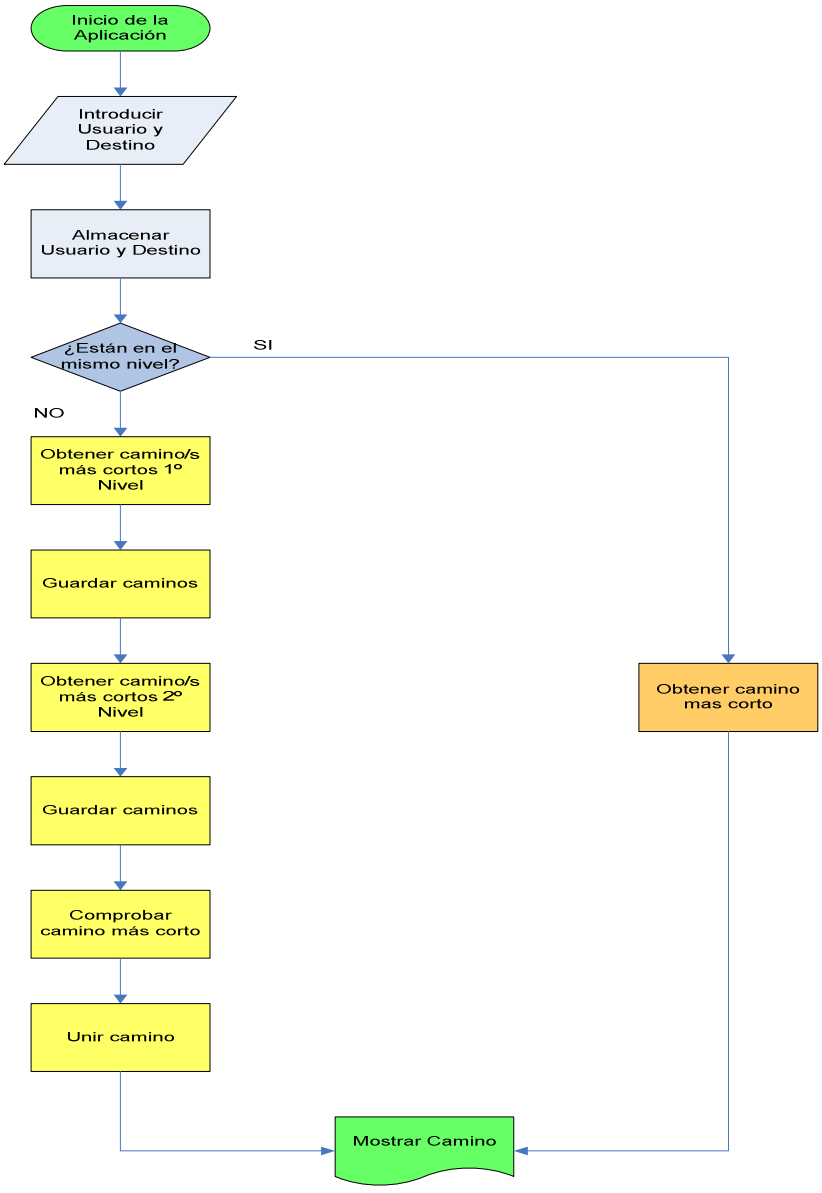


Imagen 29. Diagrama de desarrollo de la aplicación

Lo que se pretende con la aplicación es mostrar el camino mas corto solicitado por un usuario. Para ello, es necesario que el usuario introduzca los datos, es decir, la aplicación necesita el nombre del usuario y su destino para poder encontrar el camino.

Una vez que los datos se han introducido, se comprobará si el usuario y el destino se encuentran en el mismo nivel. Ya que dependiendo del nivel se realizarán unos pasos u otros.

Si el usuario y el destino están en el mismo nivel, se obtendrá automáticamente el camino más corto para llegar al destino. La búsqueda se basa en el algoritmo de Dijkstra [10] que será explicado en detalle mas adelante. También, se tiene en cuenta que si hay más de un destino (por ejemplo, el usuario quiere ir a una tienda de deportes y hay más de una en la base de datos) siempre se obtendrá el camino más corto al destino que más cerca esté del usuario. Y por último, se mostrará el camino.

Sin embargo, si el usuario no se encuentra en el mismo nivel que el destino, los pasos a seguir siempre se hacen en un sentido determinado. Es decir, independientemente de si el usuario se encuentra en el primer nivel o en el segundo nivel, siempre se comenzará por el primer nivel.

Esta decisión se ha toma así porque al pensar como proceder cuando se llega a este caso, se vió que los pasos eran los mismos solo que al revés. Es decir, si el usuario esta en el primer nivel y quiere ir a un punto del segundo nivel, primero hay que hallar el camino para llegar al segundo nivel y después hallar el camino dentro del segundo nivel hasta llegar al punto deseado. Y si el usuario está en el segundo nivel hay que hallar el camino hasta un nodo cuya distancia a un vértice del padre (que se representa como un polígono) se mínima y después hallar el camino desde ese vértice (es decir, desde el padre) hasta el punto deseado.
Aunque como se ha indicado antes los pasos son los mismos, este último caso era más complicado de desarrollar ya que a la vez que se va obteniendo el camino habría que ir calculando la distancia a un vértice del padre, es por eso que se tomo la decisión de comenzar siempre por el primer nivel.

Por tanto los pasos a seguir son los siguientes:

1. Obtener el camino o caminos (cabe la posibilidad de que se pueda llegar por varios caminos a un mismo sitio) más corto dentro del primer nivel.
2. Una vez que se han obtenido los caminos se guardan para utilizarlos más adelante.
3. Obtener el camino/caminos (cabe la posibilidad de que existan nodos en el segundo nivel con el mismo nombre, es decir, si el usuario quiere ir a un concesionario de coches puede existir mas de uno en la base de datos) más cortos dentro del segundo nivel. En este caso el nodo inicio es un nodo del segundo nivel cuya distancia es la más pequeña a un vértice del padre (el cálculo de esta distancia se explica más delante) y el nodo fin sería el destino o el usuario dependiento de la situación en la que nos encontremos.
4. Guardar los caminos obtenidos en éste segundo nivel.
5. Comprobar cual es el camino más corto uniendo todas las posibilidades viables.
6. Unir el camino más corto. Es decir, unir el camino del primer nivel con el camino del segundo nivel que ha resultado ser el más corto.
7. Mostrar el camino completo.

Finalmente, usuario ya tiene el camino que buscaba y podrá llegar a su destino fácilmente.

En las Imágenes 30 y 31 se ve mejor éstos pasos:

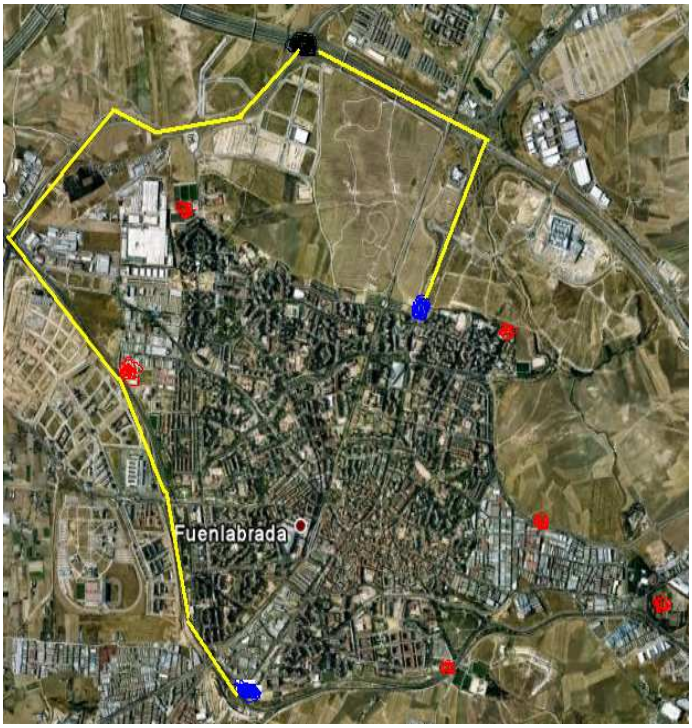


Imagen 30. Ejemplo del primer nivel de la jerarquía

En la Imagen 30 se observa que es la ciudad de Fuenlabrada situada en el primer nivel de la jerarquía y que va a ser representada como un nodo cuya forma geométrica es un polígono. Por tanto, los vértices de éste polígono son los puntos de color rojo y azul. Además, los puntos de color azul son los accesos para entrar dentro de la ciudad, es decir, para acceder al segundo nivel. El usuario está situado en el punto negro, y las líneas de color amarillo representan los caminos por los que el usuario puede ir para acceder al segundo nivel.

Imagen 31. Ejemplo del segundo nivel de la jerarquía

En la Imagen 31 se observan las calles y puntos de interés de Fuenlabrada situados en el segundo nivel de la jerarquía. Estos puntos de interés están representados de color verde, y el punto de color rosa es el destino del usuario. Además, también se pueden observar dos puntos azules que representan dos de los vértices del nodo padre (son los que aparecen en la Imagen 30). Siguiendo los pasos anteriores hay que calcular la distancia de cada uno de los nodos (puntos verdes) a los puntos azules y en cada caso coger la más pequeña para seguidamente calcular los caminos representados por las líneas amarillas.
De entre todos los caminos posibles se cogerá el más corto y se mostrará al usuario.

- Base de datos: es el componente más importante de este sistema, ya que es la encargada de gestionar los datos espaciales que tiene almacenados. A continuación, se describe su diseño con más detalle.

3.2 Base de Datos

Antes de implementar una base de datos es importante realizar un correcto análisis de la información que se quiere almacenar en ella, con la consecución del consiguiente diseño E/R. Para llevar a cabo esto, se irá identificando que información almacenar y se le asociara un subesquema E/R hasta haber finalizado el estudio completo del problema, obteniendo finalmente el esquema E/R.

La elaboración de un esquema E/R que recoja la semántica de un determinado Universo del Discurso es un proceso creativo para el que no existe un procedimiento definido. Sin embargo, sí es posible seguir una serie de recomendaciones o heurísticas que ayuden en el diseño. Estas recomendaciones no son reglas que siempre funcionen sino que en algunos casos son adecuadas y en otros no.

A continuación se exponen brevemente cuáles son los constructores de los que consta el modelo E/R:

- Entidades: son los objetos principales sobre los que debe recogerse información y denotan personas, lugares, cosas o eventos de interés. Aparecen reflejadas en los problemas como nombres. A cada una de las posibles ocurrencias de la entidad se le denomina ejemplar.
- Atributos: Se utilizan para detallar las entidades asignándoles propiedades descriptivas tales como nombre, color y peso. Existen dos tipos de atributos: identificadores y descriptores. Los primeros se utilizan para distinguir de manera única cada una de las ocurrencias de una entidad, mientras que los descriptores se utilizan para describir una ocurrencia de una entidad. También es posible especificar atributos en las interrelaciones. Los atributos también aparecen reflejados en los problemas como nombre.
  Además, también es posible recoger otras restricciones semánticas sobre los atributos como Identificadores Principales y Alternativos, atributos obligatorios/opcionales (si un atributo debe tomar o no un valor), atributos univaluados/multivaluados (si un atributo toma un único valor o varios), atributos derivados (si un valor se obtiene a partir de otros elementos del esquema E/R), atributos compuestos/simples (dependiendo de si un atributo es o no un agregado de otros atributos).

  Las entidades pueden clasificarse por la fuerza de sus atributos identificadores, es decir, por su dependencia o no dependencia respecto a otras entidades. Las entidades fuertes tienen existencia propia, es decir, poseen identificadores internos que determinan de manera única la existencia de sus ocurrencias. Las entidades débiles pueden serlo por dos motivos: bien porque su existencia en la base de datos depende de una entidad fuerte, bien porque requieran para su

identificación de los atributos identificadores de una entidad. En el primero de los casos se habla de Dependencia en Existencia y en el segundo de Dependencia en Identificación.

- Interrelaciones: representan asociaciones del mundo real entre una o más entidades. Se caracterizan por su nombre, el grado (numero de entidades que participan en la interrelación), el tipo de correspondencia (numero máximo de ejemplares de una entidad asociados a una combinación de ejemplares de las otras entidades en l interrelación, que puede ser 1 ó N). Se denomina ejemplar de la interrelación a cada combinación de ejemplares de las entidades interrelacionadas que constituyen una ocurrencia en la interrelación.

  Se definen cardinalidades máximas y mínimas de las entidades que participan en una interrelación como el número máximo y mínimo de ejemplares de una entidad que puede relacionarse con un único ejemplar de la otra, u otras entidades que participan en la interrelación. Las restricciones de cardinalidad se representan por una etiqueta, (0,1), (1,1), (0,N) o (1,N), situada en la línea que conecta la entidad con el rombo que representa el tipo de interrelación.

  En los problemas, la aparición de verbos podrá indicarnos, en algunos casos, la existencia de una interrelación en el esquema E/R.

  En cuanto a las generalizaciones, proporcionan un mecanismo de abstracción que permite especializar una entidad (que se denominará supertipo) en subtipos, o lo que es igual, generalizar los subtipos en el supertipo.
  Se pueden identificar generalizaciones si encontramos una serie de atributos comunes a un conjunto de entidades. Estos atributos comunes describirán al supertipo y los atributos particulares permanecerán en los subtipos. Puede ocurrir que los subtipos no tengan atributos propios; en ese caso, solo existirán subtipos.

  Existen otras restricciones semánticas relacionadas con las especializaciones como son la totalidad/parcialidad la exclusividad/solapamiento. Si las ocurrencias de los subtipos de una generalización cubren al supertipo (es decir, no hay ocurrencias en el supertipo que no pertenezcan a ninguno de los subtipos) entonces se trata de una generalización total y en caso contrario parcial. Por otro lado, si puede haber ocurrencias que pertenezcan a más de uno de los subtipos entonces se trata de generalizaciones con solapamiento; en caso de que los subtipos sean disjuntos se habla de generalizaciones exclusivas.

Una vez que se ha definido el esquema E/R así como todos sus componentes, se pasa a llevar a cabo la creación del esquema E/R que recogería toda la información del problema que se intenta resolver de una forma clara y sencilla. Para ello, para recoger toda la información, se van a seguir los siguientes pasos:

1. Identificación de entidades.
2. Asignación de atributos a las entidades.
3. Interrelaciones entre entidades.

3.2.1 Identificación de entidades

Las entidades son los objetos principales sobre los que debe recogerse información. En este problema, se detectan como posibles entidades:

- *Usuario*: La aplicación desarrollada va a ser usada por personas identificadas como usuarios, y de cada una de ellas va a ser necesario recoger cierto tipo de información.

- *Nodo*: En este caso, es una entidad esencial ya que será el lugar de destino al que querrá llegar el usuario, así como los hitos espaciales que tiene que ir alcanzando a lo largo del camino calculado. Dependiendo del nivel jerárquico en el que se encuentre el nodo representará un lugar u otro. Por ejemplo, si está en el nivel 0 representará salidas de las carreteras, entradas a ciudades, etc. y si está en el nivel 1 representará bancos, farmacias, casas, etc.
  Esta entidad, almacena todos los puntos definidos sobre la red.

- *Enlace*: Representa los enlaces de la red, es decir, la línea o curva que une dos nodos. Esta entidad se utiliza para poder guiar al usuario cuando éste solicite un lugar determinado. Por ejemplo, en el nivel 0, representará tramos de carretera, mientras que en el nivel 1 representará tramos de calles.

- *Camino*: Un usuario solicitará puntos de interés, y sería sumamente útil almacenar los caminos ya encontrados, pues si en un futuro se pregunta por el mismo camino que uno calculado en el pasado (por el mismo usuario o por otro distinto), no será necesario realizar de nuevo la búsqueda, solo se tendrá que mostrar el camino que anteriormente se almacenó. Por ello, es necesario crear una entidad encargada de dicho almacenamiento, asociando a cada camino los nodos de inicio y fin de dicho camino.

Esta última entidad es opcional, pero para este caso se considera necesaria debido a la utilidad de almacenar los caminos para llegar a los lugares solicitados por los usuarios.

3.2.2 Asignación de atributos a las entidades

Una vez extraídas todas las entidades necesarias para modelar la base de datos, se identificarán los atributos para cada una de ellas. Es decir, se detallarán sus propiedades descriptivas.

Estas propiedades son: identificadores y descriptores. Los primeros se utilizan para distinguir de manera única cada una de las ocurrencias de una entidad, mientras que los descriptores se utilizan para describir una ocurrencia de entidad.

A continuación, se detalla cada entidad con sus atributos.

- Entidad **Usuario**: Identifica a cada uno de los usuarios de la aplicación. Posee los atributos:

  - *Nombre_completo*: Almacena el nombre del usuario. Es el identificador principal y por tanto es único. No pueden existir dos usuarios con el mismo nombre.

- Entidad **Nodo**: Identifica cada uno de los puntos de interés que se almacenarán en la red. Posee los atributos:

    - *id_nodo*: Es el identificador del nodo en la red, es decir, clave primaria. Así pues, es obligatorio, único y está formado por caracteres numéricos, permitiendo así diferenciar un nodo de otro nodo.
    - *coordenadas_nodo*: Contiene las coordenadas del nodo, longitud y latitud, ya que es una red espacial con validez sobre la superficie terrestre (sistema de referencia espacial georeferenciado. También es obligatorio y único, ya que no pueden existir dos nodos con las mismas coordenadas.
    - *coste_nodo*: Contiene el coste de atravesar ese nodo, es decir, lo que cuesta atravesarle. Por ejemplo, si el usuario presentase algún tipo de discapacidad motriz, el nodo que representase unas escaleras tendría *coste_nodo* igual a infinito para él. Para la representación de este trabajo se considera el *coste_nodo* de todos los nodos de la red igual a 0, ya que no se ha tenido en cuenta por ejemplo las discapacidades en los usuarios, obstáculos, etc.
    - *nivel_nodo*: Ya que se está trabajando con una red jerárquica, este atributo contiene el nivel en el que está el nodo. Por ejemplo, si el nodo es una incorporación a una carretera (es decir, pertenece al nivel mas alto de la jerarquía) tendrá asignado un 0 en el *nivel_nodo*, mientras que si el nodo es una farmacia (información representada en el nivel mas bajo de la jerarquía) el valor del *nivel_nodo* será 1.

      Para el desarrollo de este trabajo solo se han considerado dos niveles. Un primer nivel que representa las carreteras que llegan y rodean al nodo padre. Y un segundo nivel que representa las calles con los puntos de interés del nodo padre. La jerarquía es necesaria ya que se va a poder trabajar con redes más pequeñas filtradas por su nivel jerárquico, de tal forma que esto va a ser posible ya que va a ver nodos asociados a un padre para poder realizar este trabajo. Si no hubiese jerarquía, toda la red estaría en un solo nivel y siempre se tendría que trabajar con toda la red.

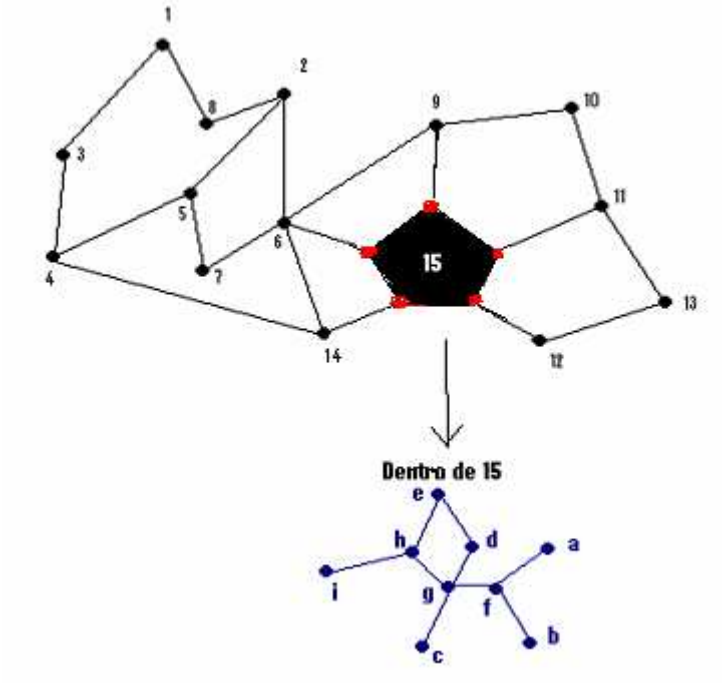


Imagen 32. Red

La Imagen 32 representa las distintas relaciones entre padres e hijos que se han llevado a cabo para desarrollar este proyecto.

- Los nodos y enlaces de color negro están en el nivel más alto. Como solo se han considerado dos niveles, dichos elementos de este nivel no tienen padre ya que se encuentran en el nivel superior. Los enlaces son considerados tramos de las carreteras y los nodos puntos de unión para enlazar unas carreteras con otras o simplemente puntos para que la representación de un enlace no sea excesivamente grande.
- Los enlaces formados por nodos de color negro y rojo representan el acceso para llegar al padre del nivel inferior, es decir, el tramo de carretera para acceder a la ciudad.
- El padre, representando por el número 15, es el punto del medio del área en lugar del polígono. Éste padre se considera como una ciudad en general, es decir, como Getafe sin entrar en detalles de calles y puntos de interés.
- Los nodos y enlaces de color azul están en el nivel inferior, y todos ellos tienen como padre el nodo representado por el número 15. Estos nodos y enlaces representas puntos de interés y calles de una ciudad.

La decisión de situar las carreteras en el nivel más alto y las calles y puntos de interés de una ciudad en el nivel más bajo a sido porque funcionalmente queda mejor, es decir, comenzar con una red grande como es la red de carreteras e ir disminuyendo a redes más pequeñas como son las calles de las ciudades, incluso una red más pequeña todavía como pueden ser los barrios de las ciudades. Esto último se refleja en una línea futura.

- *nombre_nodo*: Contiene el nombre del nodo. Se puede dar el caso de que existan varios nodos con el mismo nombre, pero no supondrá un problema ya que se distinguirán por el *id_nodo* y las *coordenadas_nodo*. Por ejemplo, puede haber una farmacia en la calle Chipre, y otra farmacia en la calle Grecia. El nombre es el mismo, pero sus coordenadas son distintas. Esto solo se aplica a los nodos del nivel más bajo. Los nombres de los nodos del nivel superior tanto si son padres como sino lo son, tendrán nombres distintos, ya que en el caso de los padres no pueden existir dos ciudades con el mismo nombre, y en el caso de nodos que no son padres por facilitar el desarrollo del proyecto.
- *tipo_nodo*: Contiene una descripción del nodo. Por ejemplo: tienda de deportes, video club, etc.

- Entidad **Enlace**: Identifica los enlaces de la red. Haciendo un análisis de la aplicación que se quiere crear, se puede deducir que esta entidad posee los siguientes atributos:

  - *id_enlace*: Es el identificador del enlace en la red, es decir, clave primaria. Así pues, es obligatorio, único y está formado por caracteres numéricos, permitiendo así diferenciar un enlace de otro enlace.
  - *coordenadas_enlace*: Contiene las coordenadas del enlace, longitud y latitud. También es obligatorio y único ya que no pueden existir dos enlaces con las mismas coordenadas.
  - *coste_enlace*: Contiene el coste de atravesar ese enlace, es decir, el coste de llegar de un nodo del extremo del enlace al nodo del otro extremo del enlace. En el caso de la aplicación que se está desarrollando, el coste indicará la distancia euclídea que existe

entre ambos nodos. Este atributo es un atributo derivado, ya que se calcula a través de los dos nodos que forman el enlace.
Dicha distancia euclídea se calcula mediante la fórmula siguiente:

$$d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

Esta distancia se calculará mediante un disparador, cada vez que se inserte una nueva fila, o se modifique una antigua, en la entidad **Enlace**.

- *nivel_enlace*: Ya que se está trabajando con una red jerárquica, este atributo contiene el nivel en el que está el enlace. Al igual que ocurría en los nodos, si por ejemplo, el enlace representa un tramo de una carretera, dicho enlace tendrá como *nivel_enlace* un 0, mientras que si representa un tramo de una calle, dicho enlace tendrá como *nivel_enlace* un 1.

- Entidad **Camino**: Almacena los distintos caminos encontrados, es decir, el usuario solicita un destino, se muestra el camino encontrado para llegar a él y a la vez se almacena en la base de datos, ya que, como se explicó anteriormente, otro usuario (o el mismo usuario) puede solicitar el mismo destino en un momento futuro. Posee los atributos:

  - *id_camino*: Es el identificador del camino en la red, es decir, la clave primaria. Así pues, es obligatorio, único, y está formado por caracteres numéricos, permitiendo así diferenciar un camino de otro camino.
  - *coste_camino*: Contiene el coste total de atravesar el camino (suma de los costes de los nodos que se atraviesan (que como se explicó anteriormente será 0) y de los enlaces que se recorren).
  - *nombre_camino*: Contiene el nombre del camino. Es un atributo opcional. Se puede almacenar con un nombre relevante o útil.
  - *tipo_camino*: Contiene una descripción del camino. Es un atributo opcional. Se puede almacenar información que se considere relevante o útil.

3.2.3 Relaciones entre entidades

Las relaciones representan asociaciones del mundo real entre una o más entidades. Se caracterizan por su nombre, el grado (número de entidades que participan en la relación), el tipo de correspondencia (número máximo de ejemplares de una entidad asociados a una combinación de ejemplares de las otras entidades en la relación, que puede ser 1 o N).

Para ampliar la semántica recogida en una relación se utiliza la restricción de cardinalidad. Se definen las cardinalidades máximas y mínimas de las entidades que participan en una relación como el número máximo y mínimo de ejemplares de una entidad que puede relacionarse con un único ejemplar de la otra u otras entidades que participan en la relación.

Las relaciones que se han obtenido en este caso, son las siguientes:

- Entre las entidades ***Usuario*** **y** ***Nodo*** hay dos relaciones:

  1. Un usuario está representado por un y sólo un nodo, por tanto la cardinalidad máxima y mínima es 1. Y un mismo nodo puede estar representando a 0 usuarios o a n usuarios, por tanto la cardinalidad mínima es 0 y la cardinalidad máxima es n.
     El tipo de correspondencia será 1:N y la interrelación se representa en la Imagen 33.

Imagen 33. Interrelación entre las entidades Usuario y Nodo

2. Se tiene que un nodo puede ser solicitado por cero o múltiples usuarios, y que un usuario puede no haber buscado todavía ningún nodo, o puede haber solicitado varios a lo largo de su registro como usuario de la aplicación.
   Por tanto, el tipo de correspondencia será N:M y la interrelación se representa en la Imagen 34.

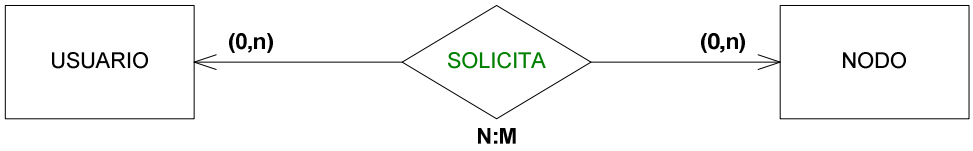


Imagen 34. Interrelación entre las entidades Usuario y Nodo

- Entre la entidad ***Nodo y Nodo***

  En este caso, se trata de una relación reflexiva, es decir, la entidad **Nodo** se relaciona consigo misma. Esto ocurre porque se puede dar el caso de que un nodo de un nivel puede ser padre de un nodo de un nivel más bajo. No quiere decir que todos los nodos tienen que tener obligatoriamente un padre.
  Por tanto, un nodo puede tener (solo uno) o no padre (cardinalidad mínima 0 y máxima 1) y un padre puede tener muchos hijos o al menos uno, de lo contrario no sería padre (cardinalidad mínima 1 y máxima N).

  El tipo de correspondencia será 1:N y la relación se representa en la Imagen 35.

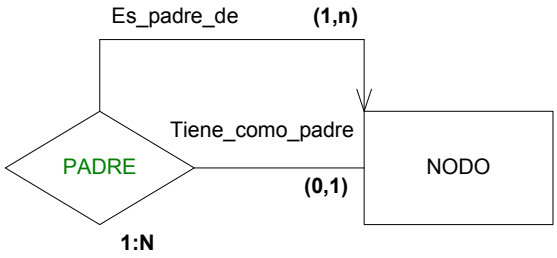


Imagen 35. Interrelación entre la entidad Nodo

- Entre las entidades ***Nodo* y *Enlace***

  En esta relación va a haber una entidad débil, la entidad **Enlace**, ya que su existencia en la base de datos depende de una entidad fuerte, en este caso de la entidad Nodo. Es decir, si no existe ningún nodo, mucho menos va a existir un enlace ya que un enlace está formado por dos nodos. Para deducir el tipo de correspondencia de esta relación hay que ver lo siguiente: un enlace tiene dos nodos, y solo dos nodos. Además, se va a tener que un mismo nodo va a poder tener

múltiples enlaces, y como mínimo tendrá uno, pues la red es conexa, y no se permitirá que un nodo esté aislado del resto de nodos de la red.
Por tanto la cardinalidad mínima para la entidad **Enlace** es 1 y la cardinalidad máxima es N. Y la entidad **Nodo**, tiene como cardinalidad mínima y máxima 2.
Esta relación es con dependencia en existencia ya que como se ha explicado antes, si no existen nodos, tampoco van a existir enlaces.

Por tanto, el tipo de correspondencia será 2:N y la relación se representa en la Imagen 36.

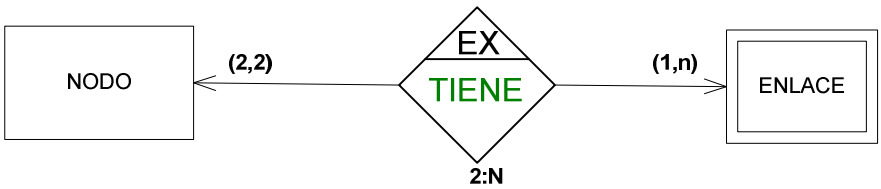


Imagen 36. Interrelación entre la entidad Nodo y Enlace

- Entre las entidades ***Nodo* y *Camino***

Para deducir el tipo de correspondencia para esta relación, hay que pensar lo siguiente: un camino siempre va a tener como mínimo dos nodos (el nodo inicio del camino, y el nodo fin del camino, que se supondrán siempre distintos, ya que no tiene sentido que un usuario solicite que le guíen al mismo lugar que en el que se encuentra) y como máximo va a tener multitud de nodos (que podría llegar a ser igual al número total de nodos en la red si el camino los recorre todos para llegar de un inicio a un fin).
Además, hay que tener en cuenta que un nodo podrá haber sido nodo inicio o fin de camino múltiples veces (por ejemplo, un punto de encuentro turístico en una ciudad suele ser un punto de inicio o fin de camino bastante usual para los turistas, de modo que se demandarán múltiples solicitudes de caminos desde/hacia él) o no haber sido usado nunca para tal fin.
Por tanto, la cardinalidad mínima para la entidad **Nodo** va a ser 2 y la cardinalidad máxima va a ser N. Respecto a la cardinalidad de **Camino**, la mínima va a ser 0 y la máxima va a ser N.

El tipo de correspondencia será 2:N y la relación resultante se representa en la Imagen 37.

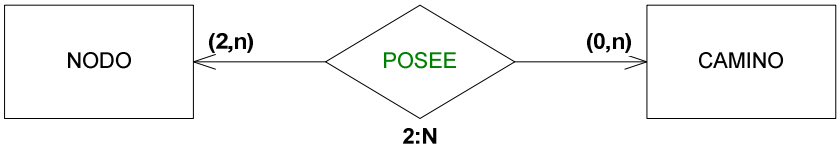


Imagen 37. Interrelación entre la entidad Nodo y Camino

- Entre las entidades ***Enlace y Camino***

Para deducir el tipo de correspondencia para esta relación, hay que pensar lo siguiente: un camino siempre va a tener como mínimo un enlace (sino no sería un camino) y como máximo va a tener multitud de enlaces.
Además, un enlace puede no formar parte de ningún camino (no se ha usado en ninguno de los caminos solicitados hasta el momento), pero de formarlo, puede estar incluido en mas de un camino. Por ejemplo, si el usuario esta en la estación de tren de Leganés Central quiere ir a la

Universidad Carlos III irá por el camino de color rojo, y si quiere ir a la comisaría de policía nacional irá por el camino de color amarillo que se puede ver en la Imagen 38.

Imagen 38. Dos caminos distintos con enlaces en común

Se puede observar que son dos caminos distintos pero con enlaces en común.
Por tanto, la cardinalidad mínima para la entidad **Enlace** va a ser 1 y la cardinalidad máxima va a ser N. Respecto a la cardinalidad de **Camino**, la mínima va a ser 0 y la máxima va a ser N. El tipo de correspondencia será N:M y la relación resultante se presenta en la Imagen 38.
Esta relación tiene, además, el atributo *sucesión_enlace_camino* que identifica la secuencia del enlace en el camino, es decir, si es el primer enlace del camino,…, o el último.

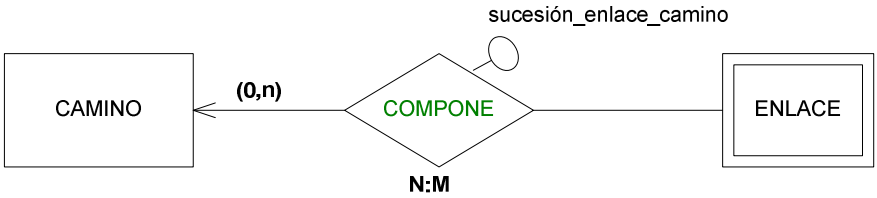


Imagen 38. Interrelación entre la entidad Enlace y Camino

3.2.4 Esquema E/R

Una vez explicadas las entidades, los atributos de las entidades, y las interrelaciones entre las entidades para crear el esquema E/R en la **Imagen 39** se muestra el esquema completo.

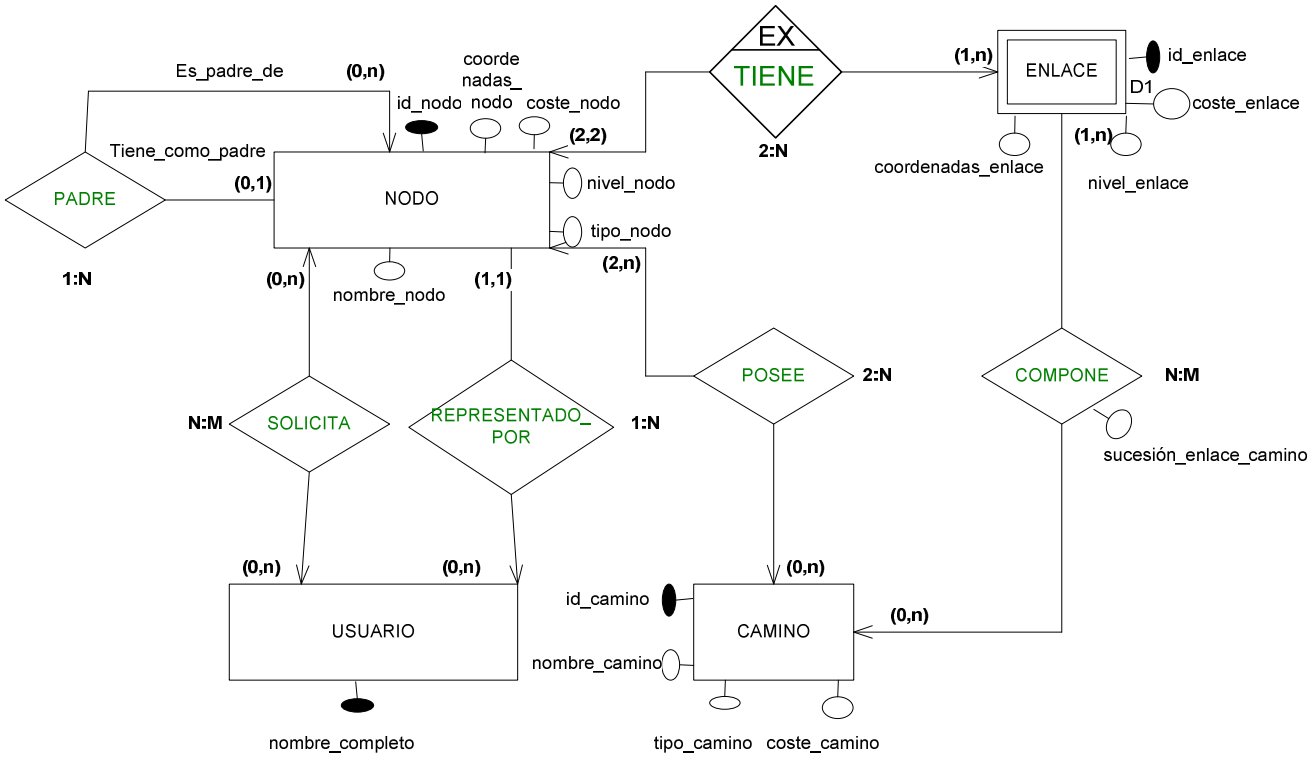


Imagen 39. Esquema E/R

3.2.5 Supuestos semánticos no reflejados en el Esquema E/R

No se puede recoger en el diagrama los siguientes supuestos semánticos:

- El atributo *nombre_nodo* de la entidad **Nodo** cuyo nivel del nodo sea el nivel más alto, y tanto para los nodos que son padre como para los que no lo son, tienen los nombres distintos. Esta ha sido la implementación que se ha tenido en cuenta, comentada anteriormente.
- El atributo derivado *coste_enlace* de la entidad **Enlace** se calcula a partir de las instancias de la entidad Nodo.

4. Implementación

En este apartado, se va a explicar como se ha desarrollado la aplicación. Para ello se describe en detalle los diferentes componentes de la misma, que en este caso son dos: una base de datos Oracle y una interfaz desarrollada en JAVA, así como la comunicación que existe entre ellos.

4.1 Repositorio de conocimiento

Siguiendo los pasos de una metodología de desarrollo de Bases de Datos, una vez realizado el esquema conceptual, se explican las decisiones tomadas para la transformación de dicho esquema al esquema relacional.

En primer lugar, hay que saber que el modelo relacional impone una serie de restricciones inherentes:

a) En una relación no puede haber dos tuplas iguales (obligatoriedad de clave primaria).
b) El orden de las tuplas y el de los atributos no es relevante.
c) Cada atributo sólo puede tomar un único valor del dominio sobre el cual está definido (no hay grupos repetitivos).
d) Ningún atributo que forme parte de la clave primaria de una relación puede tomar un valor nulo (regla de integridad de entidad).

En segundo lugar, los mecanismos que proporciona el modelo relacional para recoger restricciones semánticas o de usuario son:

a) La restricción de clave primaria (PRIMARY KEY) permite declarar un atributo o conjunto de atributos como clave primaria de una relación que identifica unívocamente cada tupla de la relación.
b) La restricción de unicidad (UNIQUE) permite definir claves de identificación alternativas (los valores de uno o varios atributos no pueden repetirse en diferentes tuplas de una relación).
c) La restricción de obligatoriedad (NOT NULL) permite declarar si uno o varios atributos de una relación deben tomar siempre un valor, es decir, no pueden tomar valores nulos.
d) La restricción de clave ajena (FOREIGN KEY), también denominada integridad referencial, se utiliza para, mediante claves ajenas (conjunto de atributos en una relación que es una clave primaria en otra o la misma relación), enlazar relaciones de una base de datos.

Por otra parte, el modelo relacional permite también definir las opciones de borrado y modificación en las claves ajenas. Estas opciones indican las acciones que hay que llevar a cabo cuando se produce un borrado o modificación de una tupla en la relación referenciada por una relación. Estas opciones son:

- Borrado/modificación sin acción (NOT ACTION): SQL permite definir la opción RESTRICT y NOT ACTION. La primera actúa como: si existen tuplas en la relación hija relacionadas con la tupla de la relación padre sobre la que se realiza la operación, entonces no se permitirá llevar a cabo dicha operación. Y la segunda es similar (pero puede diferirse su comprobación). Por lo que solo se impide la operación si al final de la misma el resultado rompe la integridad referencial.

- Borrado/modificación en cascada (CASCADE): el borrado (o modificación) de una tupla en la relación padre ocasiona un borrado (o modificación) de todas la tuplas relacionadas en la relación hija (tuplas cuya clave ajena coincida con el valor de la clave primaria de la tupla eliminada o modificada en la relación padre).
- Borrado/modificación con puesta a nulos (SET NULL): permite poner el valor de la clave ajena referenciada a NULL cuando se produce el borrado o modificación de una tupla en la relación padre. Esta opción está permitida siempre y cuando la clave ajena admita nulos.
- Borrado/modificación con puesta a un valor por defecto (SET DEFAULT): su funcionamiento es similar al caso anterior, con la excepción de que el valor al que se ponen las claves ajenas referenciadas es un valor pro defecto que se habrá especificado en la definición de la tabla correspondiente.

Una vez entendido todo esto, solo falta indicar las tres reglas básicas que se van a emplear para transformar el esquema conceptual E/R en un esquema relacional. Estas reglas son:

1. Toda entidad se transforma en una relación.
2. Las interrelaciones N:M se transforman en una relación.
3. Las interrelaciones 1:N dan lugar o bien a una propagación de clave o bien a una relación.

4.1.2 Transformación del esquema E/R al esquema relacional

Se van aplicar las reglas explicadas anteriormente para comenzar con la transformación del esquema E/R al esquema relacional.

A continuación se describen las decisiones tomadas para realizar dicha transformación:

1. Se transformará la entidad **Nodo** y la interrelación reflexiva asociada a ella. La entidad se transformará en una relación con clave primaria el identificador del nodo (*id_nodo*). Como la interrelación **Padre** es 1:N, se puede transformar en una nueva relación o se puede realizar una propagación de clave. En este caso se ha optado por realizar una propagación de clave ya que no tiene atributos propios. Por tanto la clave que se propagará a la relación NODO será el identificador del nodo padre (*padre_nodo*).
   La Imagen 40 muestra la transformación para esta entidad.

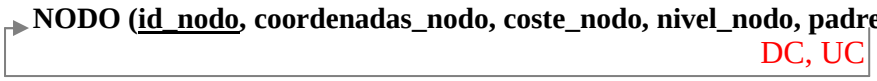


Imagen 40. Transformación de la entidad NODO

2. Se transformará la entidad **Usuario** en una nueva relación con clave primaria el identificador del usuario (*nombre_completo*) ya que no puede existir más de un usuario con el mismo nombre y apellidos. La interrelación **Representado Por** es 1:N, se ha tomado la decisión de propagación de clave, por lo que esta relación tendrá, además, el atributo *posición* como clave ajena que referencia a la relación NODO, y las opciones de borrado y modificación serán en cascada, ya que si el nodo en el que está situado el usuario no existe, no tiene sentido que su posición sea la de un punto inexistente.
   Como la interrelación **Solicita** es N:M, se debería transformar en una nueva relación o realizar una propagación de clave, pero se ha tomado la decisión de que no es necesario almacenar los nodos que son solicitados por los usuarios ya que no es relevante para el desarrollo del proyecto.

La Imagen 41 muestra dicha transformación.

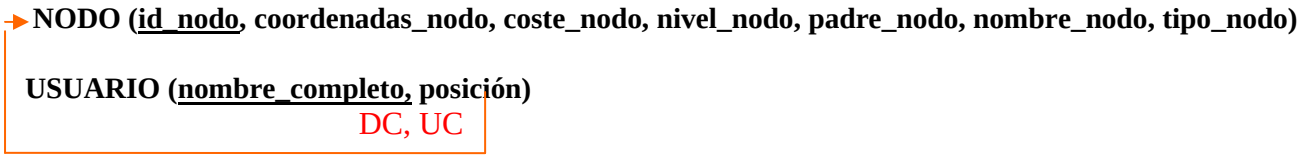


Imagen 41. Transformación de la entidad USUARIO

Se transformará la entidad débil **Enlace** en una nueva relación, con clave primaria el identificador del enlace (*id_enlace*). Como la interrelación en existencia **Tiene** es 2:N, se realiza una propagación de claves. Por tanto, la clave primaria de la relación NODO pasa a la relación ENLACE como clave ajena. Esta clave ajena esta formada por los atributos (*id_nodo_inicio* e *id_nodo_fin*) ya que un enlace consta de un nodo inicio y un nodo fin, y además, a su vez es clave alternativa ya que no pueden existir dos enlaces que unen a los mismos nodos ya que se trataría del mismo enlace. Por provenir de la transformación de una interrelación en existencia las opciones de borrado y modificación serán en cascada.
La Imagen 42 muestra dicha transformación.

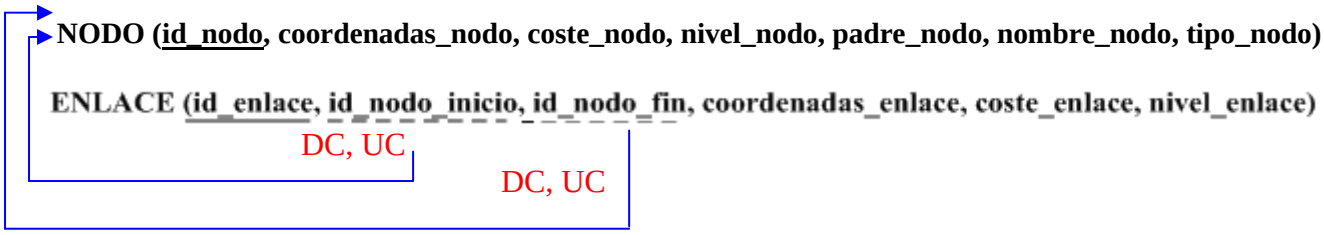


Imagen 42. Transformación de la entidad ENLACE

3. Se transformará la entidad **Camino** en una nueva relación, con clave primaria el identificador del camino (*id_camino*). Como la interrelación entre la entidad **Nodo** y la entidad **Camino** es 2:N, se procederá a realizar una propagación de claves. En este caso, se propaga la clave primaria de la relación NODO a la relación CAMINO, siendo clave ajena en esta última relación. Por tanto, las opciones de borrado y modificación serán en cascada ya que si no existen los nodos no tiene sentido que el camino siga almacenado. Estas claves ajenas son *id_nodo_inicio* e *id_nodo_fin* que en esta relación serán claves alternativas, ya que no pueden existir dos caminos cuyo inicio y fin sea el mismo.
   La Imagen 43 muestra dicha transformación.

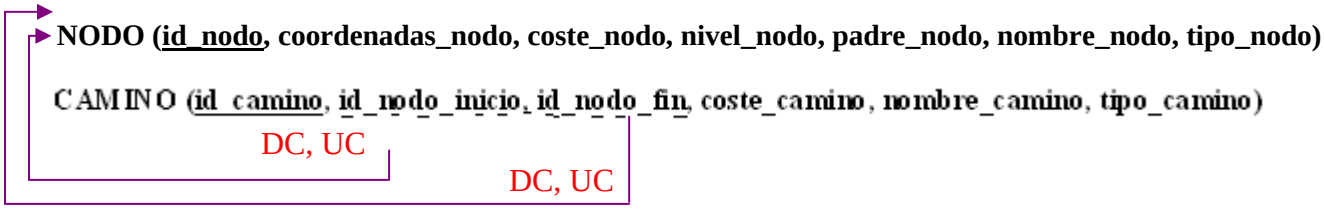


Imagen 43. Transformación de la entidad CAMINO

4. La relación COMPONE entre las relaciones CAMINO y ENLACE es N:M, por tanto se transforma en una nueva entidad ENLACE_CAMINO y se procederá a realizar una propagación de claves. Se propagará la clave primaria de la relación CAMINO y la clave primaria de la relación ENLACE. Estas dos claves serán también claves primarias en esta nueva relación, ya que no puede existir un camino que tenga dos veces el mismo enlace, por lo que las opciones de borrado y modificación serán también es cascada. Al igual que en los casos anteriores, si no existe el enlace ni el camino, no tiene sentido que la tupla esté almacenada.
   La Imagen 44 muestra dicha transformación.

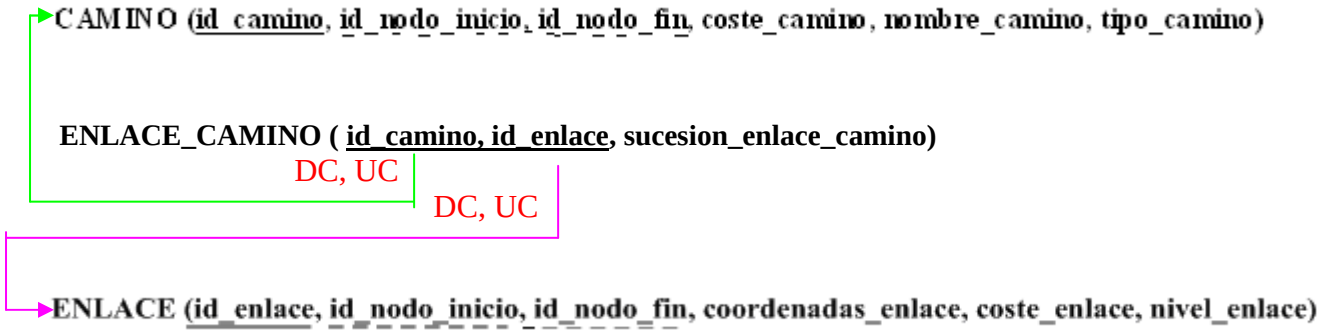


Imagen 44. Transformación de la interrelación COMPONE

En la Imagen 45 se muestra el esquema relacional con el resultado de todas estas transformaciones.

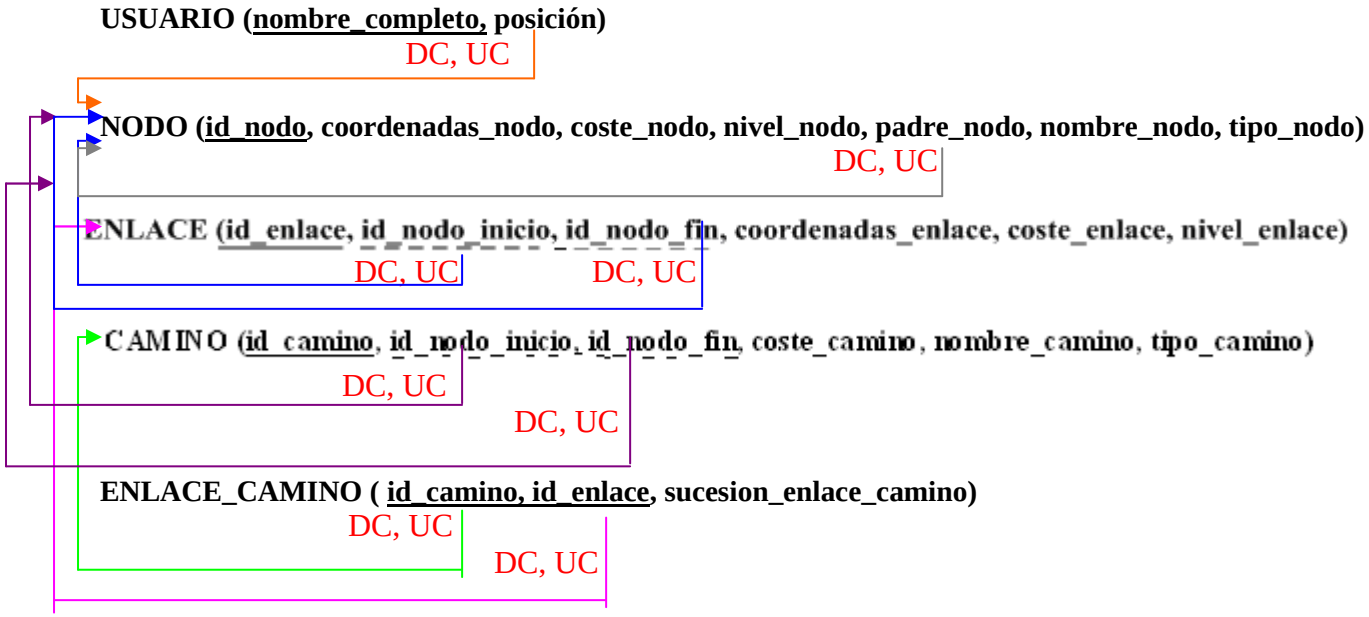


Imagen 45. Esquema Relacional

Una vez realizado el esquema relacional queda indicar como transformar el atributo derivado D1 (*coste_enlace* de la relación ENLACE). Para calcular el valor de dicho atributo, se utiliza la fórmula comentada en el apartado de "Análisis y Diseño", y se calculará mediante la utilización de una aserción que se explicará en detalle más adelante.

4.1.3 Fase lógico específico

En este apartado se va a explicar como se ha implementado cada una de las entidades, y sus atributos correspondientes, en el SGBD de Oracle.

- Los atributos que son numéricos, como por ejemplo *id_nodo*, *id_nodo_fin*, etc. se han definido como tipo NUMBER.
- Los atributos que son caracteres, como por ejemplo *nombre_completo*, *nombre_nodo*, etc. se han definido como tipo VARCHAR2.
- El atributo *nombre_nodo* es de tipo carácter y al principio de la implementación se definió también como tipo VARCHAR2, pero después se tuvo que cambiar al tipo NVARCHAR2 ya que el tipo carácter en java (String) no aceptaba la definición de este atributo como VARCHAR2.
- Los atributos *coordenadas_nodo* y *coordenadas_enlace* son de tipo SDO_GEOMETRY.
  Para insertar las coordenadas de los nodos no padre en la base de datos se hace como muestra el Ejemplo 43.

```
insert into nodo ( id_nodo, coordenadas_nodo, coste_nodo, nivel_nodo, padre_nodo, nombre_nodo,
tipo_nodo)
values (0,MDSYS.SDO_GEOMETRY(2001, 8307, MDSYS.SDO_POINT_TYPE (-3.8052,
40.2746, NULL), NULL, NULL),0, 1, 100, 'Punto 15', 'UNION');
```

Ejemplo 43. Inserción de un nodo en la base de datos

El valor 2001 indica que es un punto, el valor 8307 indica que pertenece al Sistema de Coordenadas Geodésico, y el siguiente parámetro son las coordenadas longitud y latitud de dicho punto.

Como se ha explicado anteriormente, se ha considerado el padre como un nodo, con la diferencia de que su geometría no es la de un punto, es la de un polígono que define el contorno del área que abarca dicho padre. Su inserción en la base de datos es por lo tanto distinta a la explicada anteriormente. Un ejemplo se muestra en Ejemplo 44.

```
insert into nodo ( id_nodo, coordenadas_nodo, coste_nodo, nivel_nodo, padre_nodo, nombre_nodo,
tipo_nodo)
values (100, MDSYS.SDO_GEOMETRY(2003, 8307, NULL,
MDSYS.SDO_ELEM_INFO_ARRAY (1, 1003, 1), MDSYS.SDO_ORDINATE_ARRAY(-
3.8156, 40.3043, -3.7892, 40.2961, -3.7810, 40.2951, -3.7823, 40.2882, -3.7699, 40.2803, -3.7879,
40.2749, -3.8058, 40.2738, -3.8174, 40.2934)), 0, 0, 0, 'Fuenlabrada', 'Ciudad');
```

Ejemplo 44. Inserción de un nodo padre en la base de datos

El valor 2003 indica que es un polígono, el valor 8307 indica que pertenece al Sistema de Coordenadas Geodésico, y el siguiente parámetro son las coordenadas longitud y latitud de dicho nodo. En este último parámetro se incluye la longitud y latitud de cada uno de los vértices que forman el polígono.

Con respecto a la inserción de las coordenadas de los enlaces en la base de datos, se hace como muestra el Ejemplo 45.

```
insert into enlace ( id_enlace, id_nodo_inicio, id_nodo_fin, coordenadas_enlace, coste_enlace,
nivel_enlace)
values (0, 0, 1,
MDSYS.SDO_GEOMETRY(2002,8307,NULL,MDSYS.SDO_ELEM_INFO_ARRAY(1,2,1),
MDSYS.SDO_ORDINATE_ARRAY(-3.8052, 40.2746, -3.8050, 40.2748)), 0, 1);
```

Ejemplo 45. Inserción de un enlace en la base de datos

El valor 2002 indica que es una línea, el valor 8307 indica que pertenece al Sistema de Coordenadas Geodésico, y el siguiente parámetro son las coordenadas longitud y latitud de los nodos que están unidos por dicho enlace, es decir, las coordenadas del nodo inicio del enlace y las coordenadas del nodo fin del enlace.

- Para insertar la longitud y latitud en los atributos *coordenadas_nodo* y *coordenadas_enlace*, se ha utilizado el *Google Earth* para obtener las coordenadas reales de los puntos a insertar en la base de datos. Dichas coordenadas se obtienen en grados, minutos y segundos, por lo que ha sido necesario pasarlo a grados ya que el campo correspondiente en el parámetro SDO_ORDINATE_ARRAY dentro del campo SDO_GEOMETRY solo admite que la longitud y la latitud sea en grados.

  Para dichos atributos espaciales, se ha indicado que están dentro del Sistema de Coordenadas Geodésico, por lo que fue necesario definir el rango de valores para la longitud y latitud, así como la tolerancia. Ambos van a tomar los mismos rangos de coordenadas y el mismo valor de tolerancia, pues se encuentran dentro de la misma superficie (el globo terráqueo). Para ello se ha utilizado la sintaxis del Ejemplo 46.

```
INSERT INTO user_sdo_geom_metadata
(TABLE_NAME,
COLUMN_NAME,
DIMINFO,
SRID)
VALUES (
'nodo',
'coordenadas_nodo',
MDSYS.SDO_DIM_ARRAY(
MDSYS.SDO_DIM_ELEMENT('Longitude', -180, 180, 10), -- 10 metros de tolerancia
MDSYS.SDO_DIM_ELEMENT('Latitude', -90, 90, 10) – 10 metros de tolerancia
),
8307 -- SRID para 'Longitud/Latitud’  Sistema de Coordenadas Geodésico
);
```

Ejemplo 46. Inserción del rango de la longitud y latitud, y valor de la tolerancia

Para la longitud se ha elegido el rango -180, 180 porque cubre todo el área que se quiere cubrir sobre la superficie de la Tierra. Lo mismo ocurre con el rango -90, 90 para la latitud. Estos rangos están en grados ya que se está trabajando con un Sistema de Coordenadas Geodésico.

Para la tolerancia, se ha elegido en los dos casos 10 metros (metros por tratarse de un Sistema de Coordenadas Geodésico). Se ha considerado que la distancia entre dos puntos va a ser como mínimo 10 metros, y en caso de que esa distancia sea menor los puntos se consideran en el mismo lugar. La elección de este valor ha sido porque no se querían tener los puntos muy alejados unos de otros.

- Para implementar la aserción comentada anteriormente, en Oracle se ha realizado un disparador ("trigger"), de modo que cada vez que se vaya a realizar una inserción en la tabla ENLACE saltará el disparador y ejecutará las instrucciones para obtener el cálculo de la distancia entre los nodos que une el enlace que se inserta, y que será introducido en el atributo *coste_enlace*.
  El disparador es el que se muestra en el Ejemplo 47.

```
CREATE OR REPLACE TRIGGER calculo_costo
BEFORE INSERT ON ENLACE
FOR EACH ROW
declare
distancia number;
inicio SDO_GEOMETRY;
fin SDO_GEOMETRY;
BEGIN
select coordenadas_nodo into inicio FROM nodo where :new.id_nodo_inicio = id_nodo;
select coordenadas_nodo into fin FROM nodo where :new.id_nodo_fin = id_nodo;
distancia := SDO_GEOM.SDO_DISTANCE(inicio,fin,0.005);
:NEW.COSTE_ENLACE := distancia;
END;
```

Ejemplo 47. Cálculo del atributo "coste_enlace"

Con todo esto ya estaría implementada la base de datos necesaria para la aplicación.

4.2 Unidades de Proceso

Primero se va a realizar un análisis de los distintos componentes que forman la arquitectura funcional mostrada en el apartado "Arquitectura" para saber las necesidades de implementación que tiene cada uno de ellos. En la Imagen 46 se puede observar que la funcionalidad del sistema global se divide en los siguientes componentes:

- **Aplicación**: Su objetivo es ejecutar tareas no interactivas (resolver la información pedida). Esto implica la necesidad de tener asociadas unas clases con capacidad de interactuar con la base de datos (BBDD).

- **Interfaz**: Va a ser la encargada de mostrar la interfaz gráfica para permitir la interacción del usuario con la aplicación.

Por ello, se crearon las clases mostradas en la Imagen 46, y que se explicarán posteriormente en detalle.

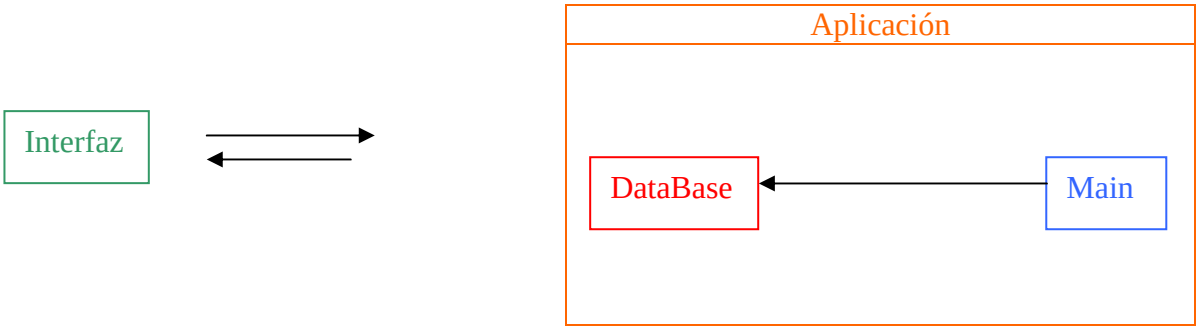


Imagen 46. Clases de la Aplicación

Para llevar a cabo esta implementación, se va hacer uso del lenguaje de programación JAVA.

Como factor muy importante, hay que destacar el método a través del cual se va a comunicar la aplicación con la base de datos. Esto va a ser posible a través de una API que tiene JAVA más conocida por sus siglas JDBC (Java Database Connectivity).

Es una API que permite la ejecución de operaciones sobre bases de datos desde el lenguaje de programación Java, independientemente del sistema operativo donde se ejecute o de la base de datos a la cual se accede, utilizando el dialecto SQL del modelo de base de datos que se utilice.

Esta conexión se detallará mas adelante.

Las ventajas siguientes son las que han llevado a utilizar Java:

- Elimina la complejidad de los lenguajes como “C” y da paso al contexto de los lenguajes modernos orientados a objetos.
- El sistema de Java maneja la memoria del ordenador por el programador. No hay que preocuparse por la memoria que no se esté utilizando, etc. Java lo realiza sin necesidad de indicárselo.
- Como el código compilado de Java (conocido como byte code) es interpretado, un programa compilado de Java puede ser utilizado por cualquier ordenador que tenga implementado el interprete de Java.
- Facilidad de crear interfaces gráficas.

Una vez aclarado el porqué de esta tecnología, se va a proceder a explicar de forma detallada cada una de las clases implementadas. La Imagen 47 muestra un pequeño diagrama de clases en el que aparecen las clases creadas con cada uno de sus métodos así como la relación entre ellas. A continuación, se analizará detenidamente la implementación realizada, explicando con más detalle cada uno de los métodos, sobre todo aquellos en los que se realizan conexiones con la base de datos.

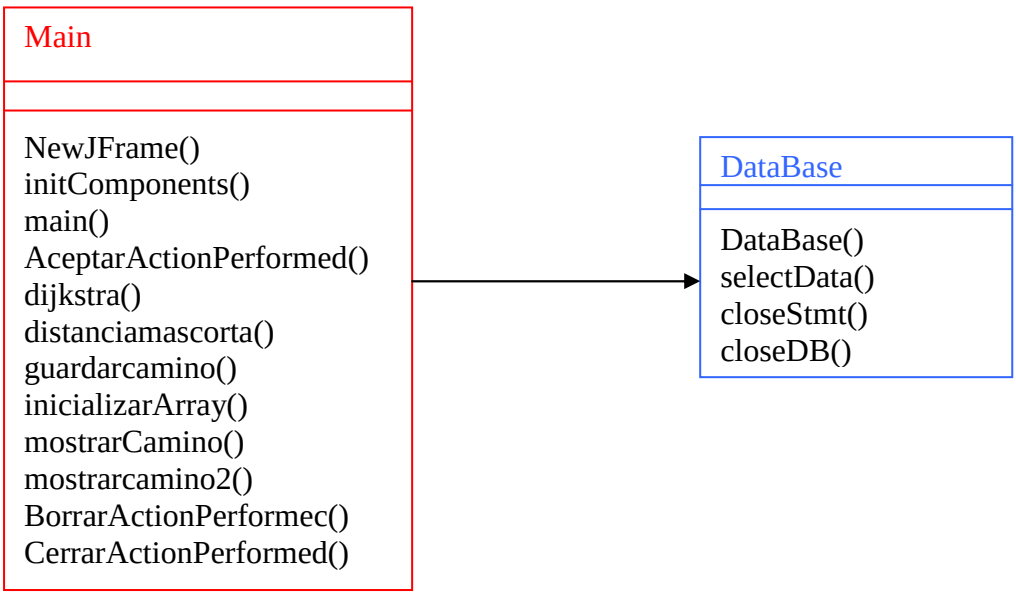


Imagen 47. Diagrama de Clases de la Aplicación

4.2.1 Main

Esta clase abarca la mayor parte de la Aplicación mostrada en la Imagen 46. Es una clase importante del sistema implementado, ya que aglutina toda la funcionalidad de la aplicación. Además, es la primera que se ejecuta en la aplicación. Esta clase encarga de crear una instancia de la clase DataBase, de mostrar la ventana con la cual el usuario interactua (Imagen 48), así como de realizar la búsqueda del camino solicitado por el usuario.

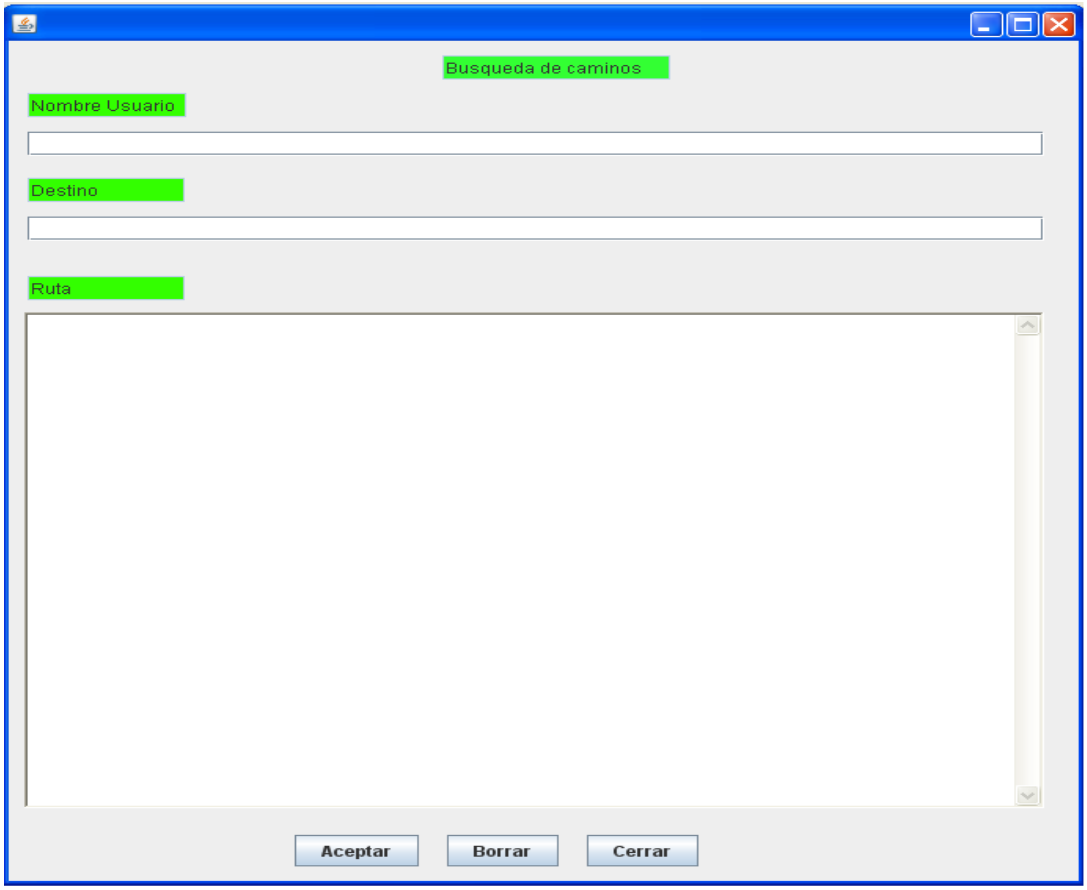


Imagen 48. Interfaz de inicio de la aplicación.

Los métodos que alberga son los siguientes:

- NewJFrame(): Este método llama al método initComponents() y crea una instancia de la clase DataBase.

- initComponents(): Inicializa todos los componentes que forman la ventana al ejecutar la aplicación.

- main(): Método que ejecuta la ventana para que sea visible por el usuario.

- AceptarActionPerformed(): Este método va a controlar toda la ejecución de la aplicación, llamando a los métodos que sean necesarios en cada momento.

  Las pautas que sigue son las siguientes:

  - Se accede a la base de datos para recuperar la posición en la que se encuentra el usuario introducido.
  - Una vez recuperada la posición, se vuelve a acceder a la base de datos para obtener los datos del nodo con el que se corresponde la posición del usuario.
  - Se obtienen los datos del nodo correspondiente al destino introducido por el usuario accediendo a la base de datos.

  Para la obtención de estos datos, se utiliza el método selectData de la clase DataBase.

- Se comprueba si el usuario y el destino están en el mismo o distinto nivel de la red.

    - Si el usuario y el destino están en el mismo nivel se procede a llamar a los siguientes métodos en este orden:

        - inicializarArray()
        - dijkstra()
        - mostrarCamino()

    El camino se mostrará en la ventana de la aplicación.

    - Si el usuario y el destino están en distinto nivel se procede de la siguiente manera:

        - Se inicializan todas las variables necesarias mediante el método inicilizarArray().
        - Se calcula el camino en el primer nivel mediante el método dijkstra(), ya que como se ha explicado antes siempre se hace en este orden en concreto independientemente de si el usuario está en el primer nivel o en el segundo nivel.
        - Se guardan los caminos obtenidos, mediante el método guardarcamino(), ya que se pueden llegar por varios nodos distintos al nodo padre.
        - Para cada uno de los nodos encontrados a través de los cuales se llega al segundo nivel, es decir, al nodo padre, se busca el vértice del padre a través del cual se accederá al siguiente nivel. Es decir, el padre se correspondende con una ciudad, por ejemplo Fuenlabrada, que está insertada en la base de datos como un nodo cuyo tipo de geometría es un poligono. Entonces mediante la siguiente función propia de Oracle se obtiene dicho vértice:

```
SDO_UTIL.GETVERTICES (geometry IN SDO_GEOMETRY)
```

          El parámetro que necesita la función en este caso, es el atributo *coordenadas_enlace,*ya que contiene el nodo a través del cual se ha llegado al nodo padre.

          Por ejemplo, hemos encontrado que a través del nodo cuyo *id_nodo* es 79 se puede llegar al nodo padre cuyo *id_nodo* es 100. Por tanto, se accederá a la base de datos para obtener el vértice del nodo padre con la ayuda de la función anterior. La consulta es la siguiente:

```
select v.X longitude, v.Y latitude FROM enlace e, TABLE (SDO_UTIL.GETVERTICES (e.coordenadas_enlace)) v WHERE id_nodo_inicio = "' + nodo + "' and id_nodo_fin = "' + padre + "' ");
```

          Esta consulta significa lo siguiente: coge las coordenadas del enlace cuyo *id_nodo_inicio* es 79 y cuyo *id_nodo_fin* es 100. Una vez que tiene las coordenadas aplica la función y guarda en las variables *longitude* y *latitude* las coordenadas del vértice del padre.

Para ser más exactos, en la base de datos va a estar almacenado el enlace que ha sido insertado de la siguiente manera:

```
insert into enlace ( id_enlace, id_nodo_inicio, id_nodo_fin, coordenadas_enlace, coste_enlace, nivel_enlace)
        values (79, 79, 100, MDSYS.SDO_GEOMETRY(2002,8307,NULL,MDSYS.SDO_ELEM_INFO_ARRAY(1,2,1),MDSYS.SDO_ORDINATE_ARRAY(-3.7833, 40.3060, -3.7892, 40.2961)), 0, 0);
```

Las coordenadas (-3.7833, 40.3060) corresponden al nodo 79, y las coordenadas (-3.7892, 40.2961) corresponde al vértice del nodo padre. Por lo que estás son las que va a devolver la función.

- Una vez obtenido el vértice, se calcula la distancia más corta de éste vértice a todos los nodos del segundo nivel mediante el método distanciamascorta() para obtener el nodo a través del cual se calculará el camino hasta el destino o el usuario dependiendo de la situación.
- Una vez obtenido el nodo, se realizan los tres primeros pasos (inicializar las variables, calcular el camino mas corto, guardar el camino). A la hora de guardar el camino, se tiene en cuenta que puede haber varios nodos con el mismo nombre (por ejemplo, varios bancos BBVA) por lo que solo se guardará el camino mas corto.

Todo este proceso desde que se terminó de trabajar con la red del primer nivel, se realiza para cada uno de los nodos a través de los cuales se puede llegar al nodo padre.

- Una vez realizado todo este proceso, se une el camino del primer nivel con el camino del segundo nivel de entre todas las posibilidades posibles para obtener el camino mas corto en combinación.

  Es decir, el usuario esta situado en un punto de la M-50 (nodo A) y quiere ir a una farmacia de Fuenlabrada entonces:
  Se cogen todos los caminos que llevan a Fuenlabrada, que resultan ser tres:

  A->B
  A->C
  A->D

  B, C y D son nodos del nivel más alto (vértices de Fuenlabrada). La farmacia más cercada a B es H, la mas cercana a C es I y la más cercana a D es H. Ya estarían los caminos en los dos niveles, ahora hay que unirlos:

  A->B->H
  A->C->I
  A->D->H

  Y el mas corto resulta ser: A->C->I, por lo tanto este se mostrará al usuario.

- Se muestra el camino mas corto mediante el método mostrarcamino2().

Todos los métodos a los que se hace referencia, se explican en detalle más adelante.

- dijkstra(): Antes de proceder a explicar en que consiste este método, hay que indicar que en un principio no iba a ser necesario. En su caso, se iba a utilizar este método propio de Oracle:

```
Path path = NetworkManager.shortestPath(Unet, p, c);
```

Lo que hace es calcular el camino más corto entre el nodo 'p' y el nodo 'c' dentro de la red 'Unet', pero no ha sido posible utilizarlo por la incompatibilidad de versiones de Oracle y Java. Además, no se tenía claro que se pudiese utilizar en redes jerárquicas.

Por tanto se ha tenido que desarrollar un algoritmo de Dijkstra [10] modificado (porque se filtra la red global por el nivel en el que se busca). El algoritmo, para su desarrollo, necesita que se le pasen los siguientes parámetros:

- El nodo inicio a partir del cual se va a buscar como llegar al resto de nodos de ese mismo nivel. Si el usuario y el destino están en el mismo nivel el nodo inicio será el nodo en el que está situado el usuario. Pero si el usuario y el destino están en distinto nivel, como se ha explicado antes se pasa dos veces por este algoritmo, por tanto la primera vez el nodo inicio será el usuario o el destino dependiendo cual esté en el primer nivel y la segunda vez el nodo inicio será aquel que se ha obtenido cuya distancia es la más corta a un vértice del padre.
- El nivel de la red en el que se van a buscar los caminos.
- Si el nivel del usuario y del destino es el mismo o no.
- El nodo padre.

Las dos últimas variables solo utilizan en el caso de que el nivel del usuario y del destino no sea el mismo.
Además, también es necesario que se almacenen los siguientes datos que serán utilizados para poder obtener el camino resultante. Estos datos son:

- Almacenar el nodo a través del cual se llega al nodo que se está visitando.
- Almacenar el coste de los enlaces (es decir, la distancia que hay para llegar al nodo que se está visitando. Por ejemplo, si se está visitando A, se encuentra que se puede llegar a A, a través de B, entonces se guarda esa distancia).
- Almacenar los nodos que se van visitando, para no volverlos a visitar.

Dicho algoritmo consiste en:

1. Mientras que no se hayan visitado todos los nodos del nivel (parámetro que se ha pasado) se tienen que realizar todos los pasos siguientes.
2. Acceder a la base de datos para obtener los nodos fin de los enlaces cuyo nodo inicio en dichos enlaces sea el nodo que se está visitando (la primera vez, será el nodo que contiene parámetro que se ha pasado).
   Y además, también se accede para obtener los nodos inicio de los enlaces cuyo nodo fin en dichos enlaces sea el nodo que se está visitando (la primera vez, será el nodo que contiene la variable pasa por parámetro).

- Con cada uno de los nodos obtenidos en ambos accesos se hace lo siguiente:

  - Si el nivel del usuario y el nivel del destino son distintos, y además el nodo obtenido en la consulta coincide con el padre, se almacena el nodo que se está visitando y la distancia (el coste del enlace). Esto se almacena para luego guardar todos los caminos desde cualquier nodo del primer nivel al padre. Por tanto, si el nivel del usuario y el nivel del destino es el mismo esto no se realiza.
  - Si el nodo obtenido en la consulta ha sido visitado, pero la distancia (coste del enlace) para llegar a él a través de otro nodo es mayor que la distancia (coste del enlace) a la que se puede llegar a él a través del nodo que se está visitando (y que no es el padre) entonces se almacena el nodo que se está visitando y la distancia (el coste del enlace) de tal forma que los datos que se habían almacenado para este nodo anteriormente quedan modificados. Sino no se guarda nada.
  - Si el nodo obtenido en la consulta no ha sido visitado y el nodo que se esta visitando no es el padre, se almacena el nodo obtenido y la distancia (coste del enlace).

3. Cuando se ha terminado de visitar un nodo, es decir, se han visto que nodos están unidos a él mediante un enlace, se procede a coger otro nodo del nivel en el que se está trabajando. Pero no se coge un nodo cualquiera, sino aquel que no haya sido visitado y en caso de que haya más de un nodo que no ha sido visitado se coge el que tenga la distancia más corta a través de la cual se ha llegado a él.

Cuando se han visitado todos los nodos, lo que se ha obtenido han sido todos los camino para llegar desde un nodo inicio a todos los nodos del nivel indicado.

- distanciamascorta(): Este método calcula la distancia que hay entre un vértice del nodo padre que se ha obtenido mediante la función SDO_UTIL.GETVERTICES explicada anteriormente (a través del cual se accederá al nivel inferior), y los nodos del nivel inferior. Para ello se utiliza la siguiente función propia de Oracle:

```
SDO_GEOM.SDO_DISTANCE (geometry1 IN SDO_GEOMETRY,
geometry2 IN SDO_GEOMETRY, tolerance IN NUMBER)
```

  Los valores geometry1 y geometry2 representan en este caso al atributo “coordenadas_nodo”.
  Este proceso se repite con cada uno de los nodos del nivel inferior, almacenando finalmente el nodo que está más cerca del vértice del padre.

- guardarcamino(): Este método, como su propio nombre indica, guarda el camino solo para el caso en el que el usuario y el destino están en distinto nivel. Si están en el mismo nivel no se utiliza este método.
  Si el camino a guardar es el correspondiente al primer nivel, se guardan todos los caminos posibles que hay para llegar al nodo padre.
  Sin embargo, si el camino a guardar se corresponde con el segundo nivel, se guardará el camino mas corto desde el nodo en el que se ha comenzado a buscar los caminos, es decir, desde el nodo más cercano al vértice del padre hasta el destino o hasta el usuario dependiendo

de cual esté en el segundo nivel. Ya que como se ha comentado anteriormente, puede haber varios nodos con el mismo nombre en la base de datos que se corresponden con el destino.

- inicializarArray(): En este método lo que se hace es crear variables de tipo vector con un tamaño determinado, crear variables de tipo array con un tamaño determinado e inicializarlas.

- mostrarCamino() y mostrarcamino2() son iguales. La única diferencia que hay entre ellos es que el primero muestra el camino cuando el usuario y el destino están en el mismo nivel, y el segundo muestra el camino cuando el usuario y el destino están en niveles diferentes.
  Se ha procedido de esta manera por una mejor facilidad para el desarrollo de la aplicación.

- BorrarActionPerformed(): Este método borra las cajas de texto de la aplicación, en el caso de que el usuario haga clic en el botón borrar para realizar una nueva búsqueda de un camino.

- CerrarActionPerformed(): Este método lo único que hace es cerrar la base de datos, y por tanto la aplicación, cuando el usuario no quiere realizar ninguna búsqueda más. Para ello se utiliza el método closeDB();

4.2.2 DataBase

Es la clase encargada de conectar la aplicación con la base de datos. Si la conexión con la base de datos falla la aplicación no funcionará, ya que todo su desarrollo se basa en constantes consultas sobre la base de datos.

Los métodos que alberga son los siguientes:

- Database(): Este método realiza la conexión con la base de datos. Para ello se utiliza la siguiente función:

  ```
  DriverManager.getConnection(url)
  ```

  En la variable “url” se indica la cadena de conexión, es decir, la dirección IP de la máquina en la que se encuentra instalada la base de datos Oracle, el usuario, y la contraseña para acceder a ella.
  Esta conexión solo se realiza una vez a lo largo de la aplicación, es decir, se abre la conexión y se realiza cualquier consulta a la base de datos. No hay que abrir la conexión cada vez que se quiera realizar una consulta.

- selectData(): Este método realiza la consulta pedida en la base de datos.

- closeStmt(): Este método cierra la instancia del objeto Statement creada. Dicha instancia se crea cada vez que se realiza una consulta a la base de datos. Sus únicos valores son verdadero o falso, dependiendo si la consulta ha sido realizada con éxito o no.

- closeDB(): Este método cierra la conexión con la base de datos. Solo se cerrará si se quiere salir de la aplicación.

5. Conclusiones y Líneas Futuras

Este apartado recoge todas las conclusiones obtenidas durante y tras el desarrollo de este proyecto, así como las posibles líneas de trabajo futuro que podrían mejorar este proyecto.

El proyecto tenía como objetivos principales:

- Realizar búsquedas en redes filtradas por su nivel dentro de la red jerárquica.
- Usar bases de datos espaciales para facilitar el trabajo de la gestión y almacenamiento de la información.

El primero de los objetivos ha quedado patente en los resultados obtenidos en la aplicación desarrollada. Ya que en todo momento se ha estado trabajando únicamente con la red específica y no con toda la red creada. La ventaja es que se ha trabado con datos reducidos, es decir, con datos filtrados por un nivel de la red, y por tanto su acceso para trabajar con ellos ha sido más rápido y nos costos.

En cuanto al uso de las bases de datos espaciales, son claras las ventajas que ha aportado a la implementación del sistema, ya que permite, de una manera sencilla, tener almacenada una red sobre la cual el usuario puede moverse, aportando herramientas de búsqueda de caminos dentro de un sistema de referencia geodésico marcado por una granularidad dada. Estas funciones, utilizando cualquier otro sistema habrían resultado muy complejas, requiriendo una aplicación muy larga y difícil de implementar.

Pero aunque el resultado ha sido satisfactorio está abierto a mejoras, necesarias algunas de ellas para solventar carencias detectadas durante la implementación o el diseño.

- Primera línea futura: La mejora más importante es compatibilizar las versiones de Oracle y Java para poder utilizar cualquier función sin ningún problema, evitando el desarrollo de algoritmos costosos como ha sido en este caso el algoritmo de Dijkstra. No obstante, hay que indicar que no se sabe si la sustitución del algoritmo de Dijkstra por el propio algoritmo de Oracle indicado anteriormente, sería capaz de realizar búsquedas en redes jerárquicas.

- Segunda línea futura: Una mejora importante (que por falta de tiempo no se ha podido incorporar en el trabajo actual) que se debería realizar es el poder realizar búsquedas de caminos independientemente de que se encuentre durante el proceso de búsqueda un nodo padre. Un ejemplo de esto se aprecia en la Imagen 49.

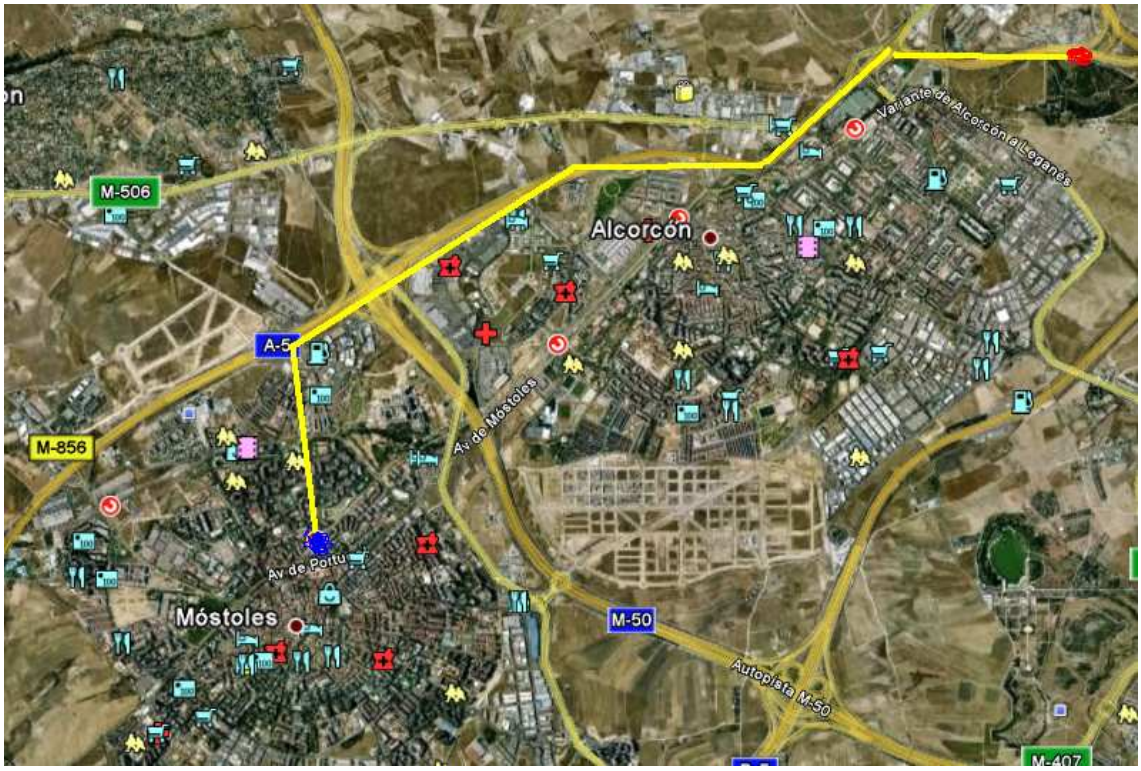


Imagen 49. Camino sin pasar dentro del padre

El usuario está en el punto rojo y quiere ir al punto azul que está dentro de la ciudad de Móstoles. Para ello tendría que seguir la ruta de color amarillo evitándose así pasar por la ciudad de Alcorcón.
Para resolver este problema, habría que buscar los dos vértices del polígono de Alcorcón y unirlos, evitando así entrar dentro de esta ciudad.
Para realizar esta búsqueda quizá se podría utilizar la función SDO_UTIL.GETVERTICES explicada anteriormente, e ir obteniendo los vértices hasta llegar al punto en el que se encuentre un enlace formado por un vértice y punto de la carretera para llegar a Móstoles, como es este caso.

Tercera línea futura: Otra mejora interesante sería la referente a la forma en la que se muestra el camino que el usuario debe seguir. Sería más intuitivo para el usuario mostrar las imágenes de las zonas (niveles) por las que tiene que ir pasando. Por ejemplo, si se mostrará una imagen parecida a la Imagen 50, el usuario podría ver en la izquierda el recorrido de la carretera por la que tiene que ir hasta llegar a la entrada de Alcorcón. Y la derecha, vería las calles por las que tiene que pasar hasta llegar a su destino (punto azul).

Imagen 50. Camino a seguir para llegar al destino

- Cuarta línea futura: Otra mejora sería detectar la posición en la que se encuentra situado el usuario que va a utilizar la aplicación tal y como lo detectan hoy en día los GPS's, o bien que fuese el usuario el que indicase donde está: "Estoy en la calle Arturo Soria en el número 1".

- Quinta línea futura: Utilizar el desarrollo realizado para otros entornos de mayor interés, o para el mismo pero cubriendo un área mayor. Para llevar a cabo esto, lo único que habría que hacer es cambiar los datos a los que se accede y añadir tantas tuplas a las tablas creadas como se necesite.

- Sexta línea futura: Una mejora más, sería introducir más niveles en la red, facilitando más las búsquedas en las redes. Por ejemplo, si se introduce un nivel que sea los barrios de las ciudades, y el usuario quiere ir a un punto que está en un determinado barrio, solo se realizaría la búsqueda en ese barrio y no en toda la ciudad a la que pertenece el punto. Con esto se reduciría aún más, los datos que se tienen que utilizar en la búsqueda del camino y a la vez menos accesos a la base de datos.

- Séptima línea futura: Al igual que se podrían introducir más niveles, también estaría bien tener más nodos padre. Para ofrecerle al usuario una mayor ayudar en la búsqueda de caminos a los que quiere ir. Porque se quiere ir a un lugar de una ciudad determinada y esa ciudad no está introducida en la base de datos no se le aporta nada, por tanto la aplicación no serviría para mucho.

- Octava línea futura: Este proyecto aún puede ampliarse más y aportar muchos valores añadidos a los que se ofrecen hoy en día. En particular, tener siempre actualizado el software sin necesidad se descargarse nuevas versiones y volviéndose a instalar. Por ejemplo, al igual que se actulizan automáticamente los navegadores como por ejemplo el Firefox si necesidad de instalarse las versiones continuamente sería muy útil para aplicaciones de este tipo.

6. Anexo

Este apartado tiene como finalidad aclarar los acrónimos que aparecen a lo largo del documento y que pueden ser desconocidos para el lector.

Algoritmo de Dijkstra [6]: También llamado algoritmo de caminos mínimos, es un algoritmo para la determinación del camino más corto dado un vértice origen al resto de vértices de un grafo dirigido y con pesos en cada arista.

API [11] (Application Programming Interface, en español interfaz de programación de aplicaciones): Conjunto de funciones y procedimientos que ofrece cierta biblioteca para ser utilizado por otro software como una capa de abstracción.

Autodesk [28]: Empresa de software que ofrece soluciones específicas para crear, gestionar y compartir activos digitales.

Bentley [29]: Ofrece soluciones para el ciclo de vida del activo de la infraestructura, adaptada a las necesidades de las distintas profesiones, - los ingenieros, arquitectos, proyectistas, constructores, fabricantes, administradores de IT, operadores e ingenieros de mantenimiento- que trabajaran en y trabajar con el activo más su vida útil. Consta de aplicaciones y servicios integrados basados en una plataforma abierta, cada solución esta diseñada para garantizar que la información fluye entre los procesos de flujo de trabajo y miembros del equipo de proyecto para permitir la interoperabilidad y la colaboración.

Bioinformática [7]: Investigación, desarrollo o aplicación de herramientas computacionales y aproximaciones para la expansión del uso de datos biológicos, médicos, conductuales o de salud, incluyendo aquellas herramientas que sirvan para adquirir, almacenar, organizar, analizar o visualizar tales datos.

CAD/CAM [8]: Es la abreviatura inglesa para las siguientes expresiones:

- Diseño asistido por ordenador (computer-aided design – CAD): Uso de un amplio rango de herramientas computacionales que asisten a ingenieros, arquitectos y a otros profesionales del diseño en sus respectivas actividades.
- Fabricación asistida por ordenador (computer-aided manufacturing – CAM): Implica el uso de computadores y tecnología de cómputo para ayudar en todas las fases de la manufactura de un producto, incluyendo la planificación del proceso y la producción, mecanizado, calendarizacion, administración y control de calidad, con una intervención del operario mínima.

CRM [9] (Customer Relationship Management): Sistemas informáticos de apoyo a la gestión de las relaciones con los clientes, a la venta y al marketing.

ERP [17] (Enterprise resource planning): Sistemas de información gerenciales que integran y manejas muchos de los negocios asociados con las operaciones de producción y de los aspectos de distribución de una compañía comprometida en la producción de bienes o servicios.

ESRI [31]: Empresa dedicada a distribuir y promocionar los Sistemas de Información Geográfica (SIG) ofreciendo todos los servicios profesionales para la implantación de un proyectas de estas características. Tecnología, consultoría, soporte técnico y formación certificada, son algunas de la muestras de distinción de ESRI España.

GeoRaster [34]: Característica de Oracle Spatial que permite almacenar, indexar, consultar, analizar y ofrecer la imagen de mapa de bits y datos cuadriculados y sus metadatos asociados.

GIS [20] (Geograhic Information System, en español Sistema de Información Geográfica): Integración organizada de hardware, software y datos geográficos diseñado para capturar, almacenar, manipular, analizar y desplegar en todas sus formas la información geográficamente referenciada con el fin de resolver problemas complejos de planificación y gestión.

GPS [21] (Global Positioning System, en español Sistema de Posicionamiento Global): Sistema global de navegación por satélite que permite determinar en todo el mundo la posición de un objeto, una persona, un vehículo o una nave, con una precisión hasta de centímetros.

Hardware [10]: Corresponde a todas las partes físicas y tangibles de una computadora: sus componentes eléctricos, electrónicos, electromecánicos y mecánicos; sus cables, gabinetes o cajas, periféricos de todo tipo y cualquier otro elemento físico involucrado.

Intergraph [32]: Empresa líder global de ingeniería y software para la gestión de información espacial (SIM), que permite a los clientes visualizar datos complejos.

JDBC [12] (Java Database Connectivity): API que permite la ejecución de operaciones sobre bases de datos desde el lenguaje de programación Java, independientemente del sistema operativo donde se ejecute o de la base de datos ala cual se accede, utilizando el dialecto SQL del modelo de base de datos que se utilice.

LRS [22](Sistema de Referencia Lineal): Es un conjunto de convenciones usadas por un observador para poder medir la posición y otras magnitudes físicas de un objeto o sistema físico en el tiempo y el espacio.

MapInfo [13]: Fue fundada en 1986 por Laszlo Bardos, Andrew Dressel, John Haller, Mike Marvin, y Sean O'Sullivan. La compañía se originó como un proyecto del Instituto Politécnico Rensselaer (RPI). El nombre original fue Navegación Tecnológica Incorporada (NTI), y el primer producto fue destinado para la navegación en automóviles.

MapQuest [14]: Editor de mapas de América y servicio Web gratuito de mapas.

NAVTEQ [33]: Líder mundial en información cartográfica digital y contenido de máxima calidad.

OCCI [36] (Oracle C++ Call Interface): Es un alto rendimiento y completa API para acceder a la base de datos de Oracle. Basado en el estándar de C ++ y el paradigma orientado a objetos, está diseñado para mejorar la productividad y la calidad en el desarrollo de aplicaciones de bases de datos Oracle.

OCI [35]**:** Es el más completo, de alto rendimiento, lenguaje nativo 'C' basado en la interfaz de la base de datos de Oracle.

OGC [15] (Open Geospatial Consortium): Fue creado en 1994 y agrupa a 372 organizaciones públicas y privadas. Las raíces del OGC se encuentran en el software fuente libre FRAS y la subsiguiente fundación OGF (Open GIS Foundation) fundada en 1992. Su fin es la definición de estándares abiertos e interoperables dentro de los Sistemas de Información Geográfica y de la World Wide Web. Persigue acuerdos entre las diferentes empresas del sector que posibiliten la interoperación de sus sistemas de geoprocesamiento y facilitar el intercambio de la información geográfica en beneficio de los usuarios. Anteriormente fue conocido como **Open GIS Consortium**.

OpenGIS [30]: Estándar internacional orientado a Sistemas de Información Geográfica, en el mercado existe una gran variedad de software OpenGis que permite procesas información Geoespacial.

Ordnance Survey [16] (OS): Agencia ejecutiva del gobierno del Reino Unido. Es la Agencia Nacional de Mapeado para Gran Bretaña, y uno de los mayores editores de mapas del mundo.

Radius [18] (Remote Authentication Dial-In User Server): Protocolo de autenticación y autorización para aplicaciones de acceso a la red o movilidad IP.

RFID [19] (Radio Frequency IDentification, en español identificación por radiofrecuencia): sistema de almacenamiento y recuperación de datos remoto que usa dispositivos denominados etiquetas, tarjetas, transponedores o tags RFID.

Skyline [38]: Proporciona herramientas software para permitir aplicaciones geoespaciales 3D. Con el apoyo de la fusión en tiempo real y la transmisión de grandes conjuntos de datos, abre estándares en una API completa, se pueden utilizar herramientas SkylineGlobe para integrar fácilmente capacidades de visualización geoespacial en 3D basado en la Web o en aplicaciones de escritorio.

Software [23]: Equipamiento lógico o soporte lógico de una computadora digital, y comprende el conjunto de componentes lógicos necesarios para hacer posible la realización de una tarea específica, en contraposición a los componentes físicos del sistema.

SQL [24]: Lenguaje declarativo de acceso a bases de datos relacionales que permite especificar diversos tipos de operaciones en éstas.

SQL*Loador [27]: Utilidad proporcionada por Oracle que permite cargar datos de un archivo a una o más tablas de la base de datos.

Tele Atlas [37]: Compañía con sede en los Países Bajos fundada en 1984 que ofrece mapas digitales y otros contenidos dinámicos para la navegación y servicios basados en ubicación, incluyendo personal y sistemas de navegación para automóviles.

Wi-Fi [25]: Sistema de envío de datos sobre redes computacionales que utiliza ondas de radio en lugar de cables.

Yahoo! [26]: Empresa global de medios con sede en Estados Unidos, cuya misión es “ser el servicio global de Internet más esencial para consumidores y negocios”. Posee un portal de Internet, un directorio Web y una serie de servicios, incluido el popular correo electrónico Yahoo!.

7. Bibliografía

[1] “10g_network_model_twp.pdf”

[2] “Aprenda Java como si estuviera en primero”. Autores: Javier Gracia de Jalón, José Ignacio Rodríguez, Iñigo Mingo, Aitor Imaz, Alfonso Brazalez, Alberto Larzabal, Jesús Calleja, Jon García.

[3] “Desarrollo de Bases de Datos: casos practices desde el análisis a la implementación” Autores: Dolores Cuadra, Elena Castro, Ana Mª Iglesisas, Paloma Martinez, Fc. Javier Calle, Cesar de Pablo, Arit. Al-Jumaily, Lourdes Moreno.

[4] “Diseño de Bases de Datos: Problemas resueltos. Autores: Adoración de Miguel, Paloma Martines, Elena Castro, José Mª Cavero, Dolores Cuadra, Ana Mª Iglesias, Carlos Nieto.

[5] Google Earth

[6] http://es.wikipedia.org/wiki/Algoritmo_de_Dijkstra

[7] http://es.wikipedia.org/wiki/Bioinform%C3%A1tica

[8] http://es.wikipedia.org/wiki/CAD/CAM

[9] http://es.wikipedia.org/wiki/Customer_relationship_management

[10] http://es.wikipedia.org/wiki/Hardware

[11] http://es.wikipedia.org/wiki/Interfaz_de_programaci%C3%B3n_de_aplicaciones

[12] http://es.wikipedia.org/wiki/Java_Database_Connectivity

[13] http://en.wikipedia.org/wiki/MapInfo

[14] http://en.wikipedia.org/wiki/MapQuest

[15] http://es.wikipedia.org/wiki/Open_Geospatial_Consortium

[16] http://es.wikipedia.org/wiki/Ordnance_Survey

[17] http://es.wikipedia.org/wiki/Planificaci%C3%B3n_de_recursos_empresariales

[18] http://es.wikipedia.org/wiki/RADIUS

[19] http://es.wikipedia.org/wiki/RFID

[20] http://es.wikipedia.org/wiki/Sistema_de_Informaci%C3%B3n_Geogr%C3%A1fica

[21] http://es.wikipedia.org/wiki/Sistema_de_posicionamiento_global

[22] http://es.wikipedia.org/wiki/Sistema_de_referencia

[23] http://es.wikipedia.org/wiki/Software

[24] http://es.wikipedia.org/wiki/SQL

[25] http://es.wikipedia.org/wiki/Wi-Fi

[26] http://es.wikipedia.org/wiki/Yahoo!

[27] http://oreilly.com/catalog/orsqlloader/chapter/ch01.html

[28] http://www.autodesk.es

[29] http://www.bentley.com

[30] http://www.cibersociedad.net/congres2006/gts/comunicacio.php?id=1032

[31] http://www.esri.es

[32] http://www.intergraph.es

[33] http://www.Navteq.com

[34]http://www.oracle.com/technology/products/spatial/pdf/11g_collateral/spatial 11g_reoraster_twp.dpf

[35] http://www.oracle.com/technology/tech/oci/index.html

[36] http://www.oracle.com/technology/tech/oci/occi/index.html

[37] http://www.teleatlas.com

[38] http://www.skylinesoft.com

[39] “Manual Java Swing.pdf”

[40] “ManualSpatial.pdf”

[41] “Pro Oracle Spatial”. Autores: Ravi Kothuri, Albert Godfrind, Euro Beinat.

[42] “Topology and Network Data Models.pdf”